Australian Government
Department of Defence
Defence Science and
Technology Organisation

Helicopter Slung-Load Simulation Toolbox for use with MATLAB®

Roger A. Stuckey

Air Operations Division
Defence Science and Technology Organisation

DSTO-TN-0855

ABSTRACT

This document outlines the Helicopter Slung Load Simulation (HSLSIM) Toolbox which is a set of utilities for the simulation, analysis and display of the flight-dynamic response of helicopters with various external load configurations within the MATLAB® software environment. Instructions and examples are provided for its operation and subsequent modifications. The helicopter studied is the CH-47D, with the load types including rectangular and cylindrical containers as well as plate and airfoil shapes.

RELEASE LIMITATION

Approved for public release

Published by

Air Operations Division
DSTO Defence Science and Technology Organisation
506 Lorimer St
Fishermans Bend, Victoria 3207 Australia

Telephone: (03) 9626 7000
Fax: (03) 9626 7999

© Commonwealth of Australia 2008
AR-014-325
August 2008

APPROVED FOR PUBLIC RELEASE

Helicopter Slung-Load Simulation Toolbox for use with MATLAB®

Executive Summary

The motivation behind this task comes from a requirement by the Australian Army to establish the safe operating limits of helicopters with externally slung loads in flight. The dynamic behaviour of slung-loads can be extremely difficult to predict and depends on such things as the load shape and density, the helicopter airspeed and control inputs as well as the sling configuration. Many incidents have been reported where the load has oscillated uncontrollably or completely diverged due to aerodynamic excitation resulting in damage and/or loss of equipment. The interaction of these elements is not well understood. Hence, a major requirement of the study was to develop a comprehensive helicopter slung-load model to provide a better understanding of the system dynamics and various effects involved. Since the initial development the model has been refined, extended and packaged together in the form of a Toolbox that is accessed from within the MATLAB® numerical computing software environment.

This document outlines the Helicopter Slung-Load Simulation (HSLSIM) software package and provides instructions and examples for its operation and subsequent modification. The graphical user interface is presented along with the various configuration dialogs and output windows generated during each stage of typical operation. These dialogs include configuration options for the helicopter, loads, control inputs and initial conditions. The three main stages of operation are: Balance, in which the load(s) are balanced in the free-stream below the fixed helicopter; Trim, where the helicopter and loads are both trimmed at the flight state specified and; Simulation, in which the helicopter–load system response to specified control inputs is simulated over the time range. The simulation may then be replayed either from within MATLAB® or in an external Virtual Reality Modelling Language (VRML) browser. As an example, the trim and simulation procedure for a CH-47D helicopter carrying a 3000L water tank is given. Using the graphical user interface, the helicopter-load system is trimmed and flown at a nominal speed (chosen as 100 ft/s) with a lateral doublet input. Then, using a simple script to automate the process, the same configuration is flown over a range of airspeeds and load masses. Two methods for presentation of the resulting data are then shown.

With the Helicopter Slung-Load Simulation Toolbox, the Australian Army will be able to gain further insight into the dynamic behaviour of various loads under different flight conditions. The user can examine the characteristics of particular helicopter-load configurations, or assess the performance of such configurations over a range of different flight states.

Contents

1. Introduction

The Helicopter Slung-Load Simulation package (HSLSIM) has been developed at the Defence Science and Technology Organisation (DSTO) as a fundamental part of Task ARM 07/038 and preceding tasks sponsored by Australian Army Aviation. In the past, the operations of helicopters carrying externally slung loads has often been limited and, in some cases, seriously hindered by stability and control problems. A program was consequently initiated within DSTO to use computer modelling and simulation to assist in defining the operational limits of the Australian Army Chinook CH-47D helicopter when carrying slung loads. The first phase in this program entailed the development of a helicopter slung-load model for simulation and analysis in order to provide a better understanding of the system dynamics and various effects involved.

HSLSIM was written in MATLAB®, a high-performance numerical computing environment that is built around matrix mathematics, and therefore amenable to dynamic modelling and simulation work. MATLAB® also provides an ideal workplace in which various analyses can be conducted following the simulation itself.

2. System Description

The following section provides a general description of the system, including the model formulation and implementation. For a more detailed description, refer to the previous report by Stuckey [1].

2.1 Model Formulation

Helicopter slung-load systems fall into a class of multibody systems approximated by two or more rigid bodies connected by massless links. The links can be considered either elastic or inelastic, although the rigid-body assumption excludes any helicopter or load elastic modes. Typically, the system is characterised by the configuration geometry, mass, inertia, and aerodynamic behaviour of both helicopter and load, as well as the elastic properties of the links.

In general terms, the system of interest consists of a single helicopter supporting one or more loads by means of some suspension. The model is comprised of n rigid bodies, with m straight-line links supporting a single force in the direction of the link. If the links are modelled as inelastic, $c \leq m$ constraints are imposed on the motion of the bodies and the system has $d = n*6 - c$ degrees-of-freedom (dof). If the links are modelled as elastic, there are $n*6$ dof.

A number of simplifying assumptions have been made in the model. These include the exclusion of cable mass, cable aerodynamics and rotor-downwash effects. Despite these limitations, the system has proven adequate for simulation studies [2] in which the low-frequency behaviour is of primary interest and the helicopter is initially trimmed in forward flight.

The simulation model used is based on the helicopter slung-load system introduced by Cicolani, *et al* [3]. In this formulation, the general system equations of motion are obtained from the Newton-Euler equations in terms of generalised coordinates and velocities. Following the explicit constraint method, which utilises d'Alembert's principle, the system is partitioned into coordinates such that the motion due to cable stretching is separated from that due to rigid-body, coupled dynamics. As a consequence, the constraint forces appear explicitly and a solution to the resultant generalised accelerations is determined by assuming a simple spring model for the cable.

It is also possible to obtain a solution to the inelastic approximation by nulling the stretching coordinates to obtain an explicit relation for the suspension forces. The result is computationally more efficient than conventional procedures and is readily integrated with the formulation for elastic suspension. Another benefit of the formulation is that it is easily applied to complex, multiple body systems, as in the software package presented. Aside from the core helicopter model, all code development has been done in the MATLAB® [4] numerical computing environment, which provides a high-performance language, amenable to modelling and simulation type work.

2.2 Implementation in MATLAB®

The Helicopter Slung-Load Simulation program HSLSIM consists of several modules, written in the MATLAB® language. These include the main script, an optimisation routine, a differential equation solution, an integration function, several flight-dynamic models, and various output and replay functions. There is also a graphical user interface for simplified control of the primary program functions. Alternatively, the simulation can be run through a main script, which generates the control inputs, configures the helicopter-load system properties (geometric and inertial), sets the initial system state, and then executes the trim and integration functions. The integration function is problem independent and based on an algorithm which combines various order Runge-Kutta formulas for the solution of ordinary differential equations. It requires an integration function tailored to the problem at hand, which provides a point solution to the differential equation. For the helicopter slung-load simulation, this function represents the core of the code and implements the aerodynamic models for both helicopter and loads.

Initially, an approximated model of the CH-47B, named HSL, and based on a set of linearised state-space representations at several airspeeds was used. This model was derived from a full nonlinear simulation model developed by the Boeing Vertol Company and later adapted for use at the NASA Ames Research Center [5] [6]. However, the model was inadequate for large manoeuvres or long simulations, so a higher fidelity model was introduced. This model, named ROTORGEN, was developed by Heffley [7] for the US Army Aeroflightdynamics Directorate under a NASA contract to Hoh Aeronautics Inc. It is described as a "minimal-complexity generic rotorcraft model" intended for manned simulation of large military helicopters and, in particular, the CH-47D Chinook tandem rotor helicopter, though it is also capable of simulating CH-53D and UH-1H helicopters. ROTORGEN implements a rotor inflow model based on Glaurt's representation of thrust, with the orientation (incidence) of the tip path plane defined by a set of flapping equations and the body forces based on a quadratic fluid-dynamics formulation, applicable to low-speed flight. The model is actually a

combination of two existing flight models: the Extended Stability Derivative (ESD) model developed for NASA, and the ROTORGEN thrust model developed for the US Army. As such, the ROTORGEN model has a modular structure, which combines several features of the original ESD model. These include a primary Flight Control System (FCS), rotor and body forces, ground effects, simple slung load dynamics, and a Stability and Control Augmentation System (SCAS). HSL was written in MATLAB®, and easily integrated into the simulation code structure. ROTORGEN, on the other hand, was written in Fortran-77 and therefore required an external (MEX) routine to interface with the MATLAB® environment and the simulation code. It appears as a compiled, Dynamic Link Library (DLL) file in the base directory.

HSLSIM also offers the choice of several generic loads including a box, cylinder, plate and airfoil shapes. The box shape was derived from a MILVAN container — a common helicopter cargo used in many commercial and military operations. The dimensions of a MILVAN container are $20 \times 8 \times 8$ ft and the mass typically varies from 4000 lb (empty) to 20000 lb (full). The aerodynamic model for the MILVAN was developed by Ronen [8] and constitutes a combination of aerodynamic models obtained from several published wind-tunnel test results, including unsteady aerodynamic effects. However, the actual model used, while comprehensive, is based purely on static data. The cylinder and plate shapes are all compiled from ESDU data sheets [9] and Hoerner [10] [11], while the airfoil – a finite-span NACA 0012 profile – is extracted from the Comprehensive Analytical Model of Rotorcraft Aerodynamics and Dynamics (CAMRAD/JA) code [12].

Normally, the simulation model is first initialised with a set of flight conditions and given a pre-determined sequence of control inputs to be used in the computation of a response. The model is then trimmed according to the specified flight conditions until an equilibrium state is reached, and then integrated over the time range to produce a set of simulation data. The data comprises the helicopter and load positions, orientations, the control inputs and geometric information regarding the helicopter-load configuration. As for most aeronautical systems [13], the positions are specified in right-handed Cartesian axes (with positive-z downward) and the orientations in Euler angles.

In addition to the trim and simulation software, a function for the replay of simulations was written. This function, REPLAY, displays rudimentary models of the helicopter and load(s) in 3 dimensions, with the controls in a separate window. A menu window allows the user to start, stop, pause and resume the simulation replay, as well as providing several options for the display.

The simulation output can also be exported to a file in the Virtual Reality Modelling Language (VRML) format. Using a VRML browser, or a WWW browser plug-in, such as the Cortona VRML plug-in [14], the simulation can then be replayed with higher detail in a similar 3-dimensional space in real time. Furthermore, the file can be compressed into a compact binary format using GZIP [15], which can still be interpreted by the VRML browser, but is small enough for fast transmission over the Internet.

3. User's Guide

3.1 Getting Started

The graphical user interface GHSLSIM provides simplified configuration and control of the helicopter-load system for trim and simulation. Many options are automated and limited from the full capability of the program, which is discussed in Section 3.7.

The program provides various configuration options, as well as controls for balance, trim and simulation stages. There are also controls for the output and display of the simulation, once run. Balance refers to the determination of an equilibrium state for the load(s) only, slung beneath a fixed helicopter. This intermediate step is performed to attain a state closer to the actual trim state than the default, initial one specified and is usually conducted prior to the trim stage. Since the helicopter is fixed during this stage, it will generally not be sufficiently trimmed following balance. Therefore, it is necessary to trim the helicopter and load(s) using the trim control, regardless of whether it has been balanced or not.

GHSLSIM can be started by executing the following command from inside the MATLAB® workspace:

```
>> ghslsim
```

If a configuration file exists, it may be loaded automatically by adding its name after the command. For example, executing:

```
>> ghslsim ghsldemo_3klt
```

will load the configuration options stored within **`ghsldemo_3klt.mat`**. If no file is specified, the default configuration file, **`ghsldefault.mat`**, is used.

3.2 Main Interface

The main interface is shown in Figure 3.1. There is a list of commands along the top menu bar as well as a set of fields and buttons that constitute the simulation configuration and control interface. The File menu commands are accessed through the top menu only, but the Configuration, Simulation and Output commands can all be accessed either through the top menu, or on the main interface panel.

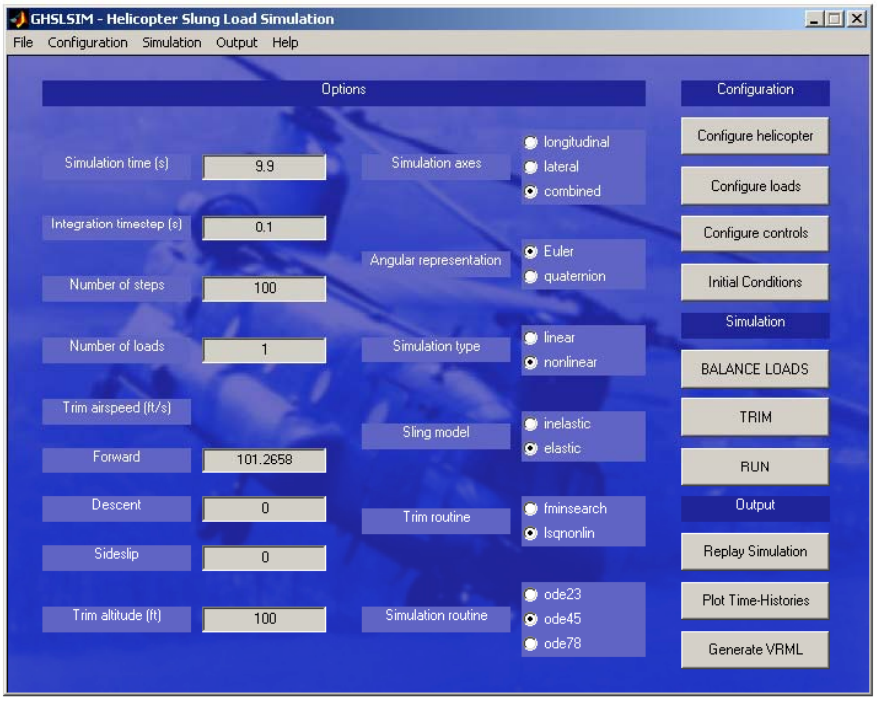

Figure 3.1: Simulation Configuration Interface

3.2.1 File Menu Commands

As well as specifying a configuration file on the command line, it is possible to load a configuration file from the file menu via the command: File → Open. This loads the configuration details and the simulation output data, if present, into the program workspace. If a file with simulation data is loaded, then the user may execute any of the *Output* commands on the main interface without having to re-run the simulation.

In order to save a configuration, or both the configuration and any simulation output data, the user simply creates a new file, or selects an existing one from the File → Save or the File → Save As menu commands. If no file has been specified, the default file, `ghsldefault.mat` in the current working directory is used.

With the Open command, the program expects a full configuration file with *all* components present, and will produce a warning otherwise. Similarly, saving to file using the Save or Save As commands saves *all* of the configuration information and simulation data, if present. If the user wishes to load or save only one component of the configuration, then they must use the import or export commands, respectively. The File → Import Config Data command will import whatever components are present in the chosen partial-configuration file. All other components will remain as they were, prior to importing. The File → Export Config Data has several sub-commands: Simulation, Helicopter, Load, Control, and Trim. These can be used to export single components from the configuration, which can later be imported as detailed above.

Preferences for the program are set in the main menu via: File → Preferences.... These include three fields for the location of a VRML-capable *Web Browser*, the *VRML Directory* and a *GZIP Executable*, as shown in Figure 3.2.

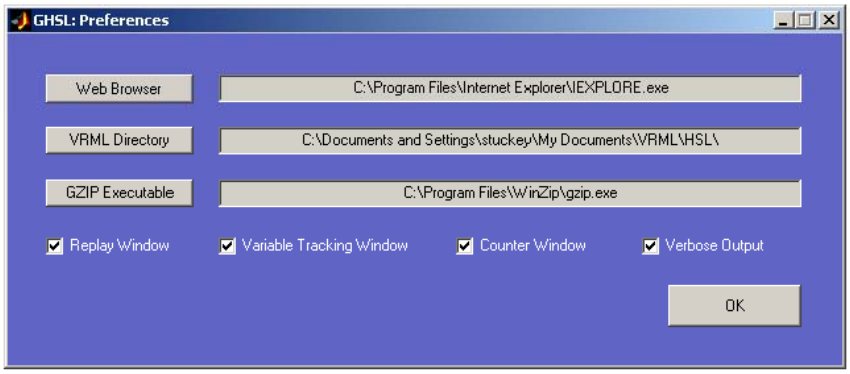

Figure 3.2: Program Preferences Interface

If not present, or incorrect, the *Web Browser* location can be typed directly into the text field, or located by simply clicking the button and manually searching through the Windows directory structure. The *VRML Directory* is the location of all helicopter and load VRML models included in the distribution, and the directory into which any generated VRML simulations will be written. This string ends with a trailing directory delimiter. If the *GZIP Executable* is installed, the last field may be filled, or located as with the *Web Browser*.

The preferences also include four checkboxes for various windows and screen output that can be activated during balance, trim, and simulation. These are the *Replay Window*, the *Variable Tracking Window*, the *Counter Window*, and the *Verbose Output* flag. The first two will be explained in more detail in Section 3.4. The *Counter Window* is just a visual indicator of the progress during simulation and post-processing stages. Checking the *Verbose Output* preference will cause the program to display numerical data during trim and simulation iteration loops.

3.2.2 Simulation Configuration

Each of the fields and options displayed in the main interface are required by the simulation. The *Simulation time* t_N, *Integration timestep* dt, and *Number of steps* N are all related via the following equation:

$$t_N = dt * N \tag{1}$$

Altering the *Simulation time* will also adjust the *Number of steps* and altering the *Integration timestep* or the *Number of steps* will adjust the *Simulation time*, such that all parameters conform to the above relationship. In general, a smaller *timestep* will yield more accurate simulation results and reduce the possibility of numerical error and subsequent divergence. For most configurations, a value between 0.05 s and 0.1 s (20 Hz and 10 Hz) should be sufficient.

The next field defines the *Number of loads*, which can be any integer between zero and three; zero referring to the helicopter-only configuration. The helicopter attachment point (hook) to which each load's slings are attached is automatically determined. For unusual sling configurations, the user must write a script, as detailed in Section 3.7.2, to construct the model and run the trim and simulation.

The following three fields set the *Trim airspeed*, in each inertial axis, at which the helicopter will be trimmed. These are (in ft/s) the *Forward* speed, rate of *Descent*, and *Sideslip*. It is not possible to trim the aircraft with a constant (non-zero) angular rate, such as in a steady turn. The last field sets the trim altitude of the helicopter, in ft.

Options presented in the interface comprise the *Simulation axes*, *Angular representation*, *Simulation type*, *Sling model*, *Trim routine* and *Simulation routine*. The *Simulation axes* define which axes are to be used in the trim and simulation. Both *longitudinal* and *lateral* axes constitute a reduced subspace in which the system is free to respond (reduced degrees-of-freedom). The *longitudinal* axes include surge, heave and pitch motion, while the *lateral* axes include heave, sway, roll and yaw motion. The *combined* axes represent a full, unconstrained system with the number of degrees-of-freedom equal to $n*6$ for an elastic sling configuration, where n denotes the number of bodies (helicopter + loads). Unless there is a specific need to excite only the longitudinal or lateral modes, it is recommended that the *combined* axes be used.

The angular displacements and rates for the system can be represented using either *Euler* angles or *quaternions*. However, this only applies to the simulation stage, as the helicopter and loads can only be balanced and trimmed using *Euler* angles. For most manoeuvres, an *Euler* representation is adequate.

If the Control System Toolbox [16] is present within MATLAB®, it is possible to simulate the system response using a *linear* approximation, which is much faster than the *nonlinear* solution. These constitute the two *Simulation types* available to the user. However, since the linearised model is obtained by computing a Jacobian at trim, it is inaccurate for anything other than very short runs. In fact, given the high nonlinearity of the coupled helicopter-load system, a *linear* simulation is really only suitable for helicopter-only configurations. If the Control System Toolbox is not present, a *nonlinear* simulation must be run.

The two *Sling models* that can be employed are *inelastic* and *elastic*, each incorporating a significantly different solution to the multi-body equations within the balance, trim and simulation stages. The *inelastic* model uses an explicitly constrained representation and therefore, has less dof than its *elastic* equivalent. Typically, the variables used in an *inelastic* model include the helicopter position and orientation, the orientation of one or more cables and the orientation of the load(s). The variables used in an *elastic* model include the position and orientation of both helicopter and load(s). Generally speaking, less dof means a lower complexity and since the only matrix inversion required for solution is calculated explicitly, it also means a faster solution. In addition, the *inelastic* model has a higher degree of orthogonality, and is consequently much better suited for trim optimisation. This will be explored in the examples that follow (Sections 3.6 & 3.7). Note that during balance and trim, it

is possible to switch between these two model representations, whilst retaining the current trim state.

Just as there are two methods for computing the system response, there are also two methods for finding a solution for the balanced and trimmed state of the helicopter-load system. If the Optimization Toolbox [17] is present within MATLAB®, a more efficient nonlinear least squares routine (*lsqnonlin*) is used. Otherwise, the basic simplex routine included with the core distribution (*fminsearch*) is used. The nonlinear least squares algorithm iteratively determines a solution based on the accelerations, or external forces and moments in each axis. The simplex algorithm, on the other hand, iteratively determines a solution based on the sum of squares of all accelerations in the model, which is why it is much less efficient. For this reason, it is highly recommended that the Optimization Toolbox be activated.

The last option within the main interface sets the order of the integration algorithm within the *Simulation routine*. This algorithm is a fixed step-size Runge-Kutta integrator and can be of 2nd/3rd order (*ode23*), 4th/5th order (*ode45*) or 7th/8th order (*ode78*). Obviously, the higher order integrators are more accurate; however, they are also more costly in terms of processing time since they make more calls to the force determination function. The lower order integrators can also cause numerical divergence due to error, particularly for stiff elastic models, so a balance must be decided upon. The default order for new simulation models is 4th/5th, which has proven adequate for most of the cases tested.

3.3 Configuration Interfaces

There are four basic configuration interfaces, which must all be set prior to simulation. These consist of the *Helicopter Configuration*, the *Load Configuration*, *Controls Configuration*, and *Trim Initialisation*.

3.3.1 Helicopter Configuration

The Helicopter Configuration panel, shown in Figure 3.3, incorporates options to select the *Helicopter type*, the *Simulation model* to be used, and the corresponding *Weight* and *Inertia*. Supported *Helicopter types* include the Bell UH-1H Iroquois, Boeing CH-47D Chinook, and Sikorsky CH-53D Sea Stallion. Both the UH-1H and the CH-53D have one hook and the CH-47D has three hooks for the carriage of external loads.

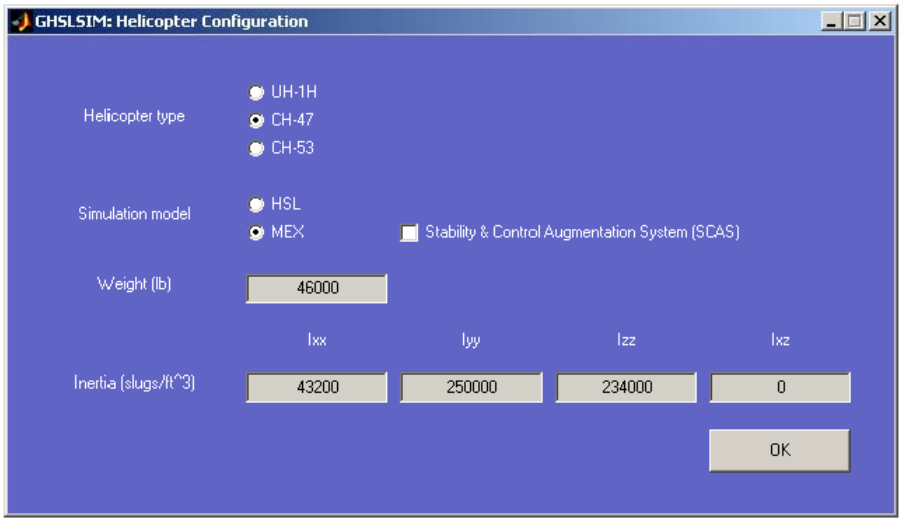

Figure 3.3: Helicopter Configuration Interface

The *Simulation model* can be selected from a simple linearised state-space model (HSL), or a full nonlinear model, based on the ROTORGEN core (MEX). As mentioned in Section 2.2, the linearised helicopter model is only an approximation and, although considerably faster, it is much less accurate than the full nonlinear model. The MEX model also incorporates a *Stability & Control Augmentation System*, which can be switched on or off via a checkbox on the panel.

Text fields for the *Weight* and *Inertia* also allow for adjustment of these attributes from within the graphical interface. The inertias include primary axes components I_{XX}, I_{YY}, I_{ZZ}, and I_{XZ}. If desired, it is possible to set any of these to infinity (inf), thereby constraining rotation in the corresponding axis. This will be explained in more detail in Section 3.7.2.

3.3.2 Load(s) Configuration

Up to three loads can be configured via the load interface, shown in Figure 3.4. For each load, there are options for the *Load type* and *Sling configuration*, as well as fields for the *Load weight*, *Load inertia*, *Size*, *Attachment* locations. There are also fields for the *Cable length*, *Stiffness* and *Damping*. *Load types* currently supported are box, cylinder and plate geometries. The box load has the most accurate aerodynamic data, based on a combination of experimental and theoretical data, whereas the cylinder and plate loads are based on theoretical data only.

The *Load weight* and *Load inertia* in each primary axis can be edited in the corresponding text boxes and, as with the helicopter, they may be set to infinity in order to constrain the rotation. Also, the size of the load can be specified, although this is merely for display purposes and has no effect on the resulting dynamics of the system.

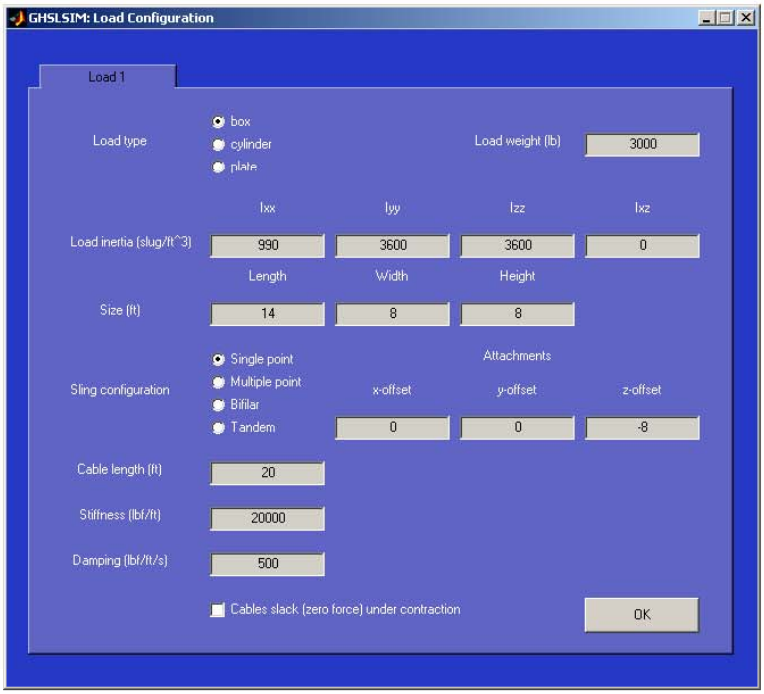

Figure 3.4: Load(s) Configuration Interface

For the *Sling configuration*, the user has a choice of four basic configurations, illustrated in Figure 3.5. These are: *Single point*, *Multiple point*, *Bifilar* and *Tandem*. Each configuration has a different number of (load) attachments, the *offsets* for which are listed next to the configuration buttons. The *Single point* configuration has one attachment, the *Multiple point* has four, the *Bifilar* has two and the *Tandem* has four as well. It is crucial to note that the order in which the attachments are listed is important. For two attachment points, the forward attachment must be listed first and the aft second. For four attachment points, the order is: forward-starboard, aft-starboard, aft-port, forward-port. Each offset is specified relative to the load axes, whose origin is located at the load cg, so most standard configurations (with attachments above the load cg) will have equal negative *z-offset* values.

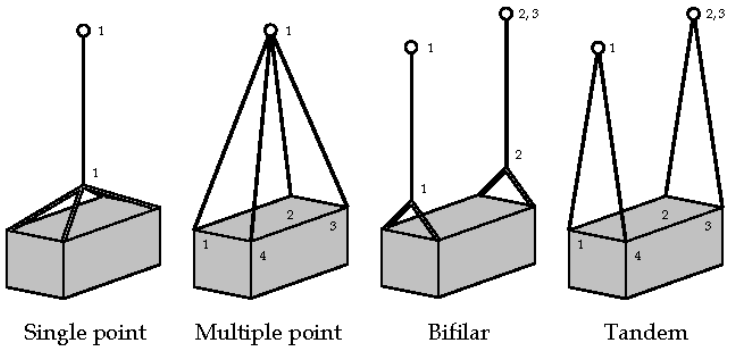

Figure 3.5: Sling Configurations

With the graphical interface, the corresponding attachment points on the helicopter are automatically determined. Table 3.1 outlines the helicopter attachment point index for various numbers of loads and each load's sling configuration. The helicopter attachment points, or hooks, are numbered from one to three in the order: forward, middle, aft. So, for example, two bifilar loads would occupy hooks (1 & 2) and (2 & 3). One tandem load would occupy hooks (1 & 3). One single point, one bifilar, and one multiple point would occupy hooks 1, (2 & 3) and 3 respectively, and so on. As mentioned previously, it is possible to define the attachment points explicitly in a script to construct the model.

Table 3.1: Helicopter Attachment Point (Hook) Index

LOAD CONFIGURATION	NUMBER OF LOADS		
	1	**2**	**3**
Single point	1	1, 3	1, 2, 3
Multiple point	1	1, 3	1, 2, 3
Bifilar	1, 3	(1, 2), (2, 3)	(1, 2), (2, 3), (2, 3)
Tandem	1, 3	(1, 2), (2, 3)	(1, 2), (2, 3), (2, 3)

As well as the sling configuration, the cable lengths and their elastic properties must be set. Using the graphical interface, the *Cable length*, *Stiffness* and *Damping* for each cable are set to the same values specified. Once again, it is possible to set these individually, as will be explained in Section 3.7.2. The last checkbox shown in the load configuration interface is to enable *Cable slack*, which imposes a zero-force when a cable is under contraction. Normally, all cables will be under tension, so this parameter will only tend to become important when the load has become unstable and diverged. Load bounce is also possible, but this is usually the result of pilot-induced oscillations, which do not occur in the simulation model. However, the main reason that this option has been left unchecked as a default is because the balance and trim optimisation routine can have difficulty in finding a solution for cables with slack. This is due to the discontinuity at the point of slack and invariable nature of the cable force when contracted. If every cable becomes slack (zero cable-force) during balance or trim, then the optimisation direction becomes indeterminable and the routine will falter.

3.3.3 Controls Configuration

The Controls Configuration panel (Figure 3.6) is slightly different than the previous two configuration panels, in that it allows the user to construct control input sequences by adding any number of basic input types.

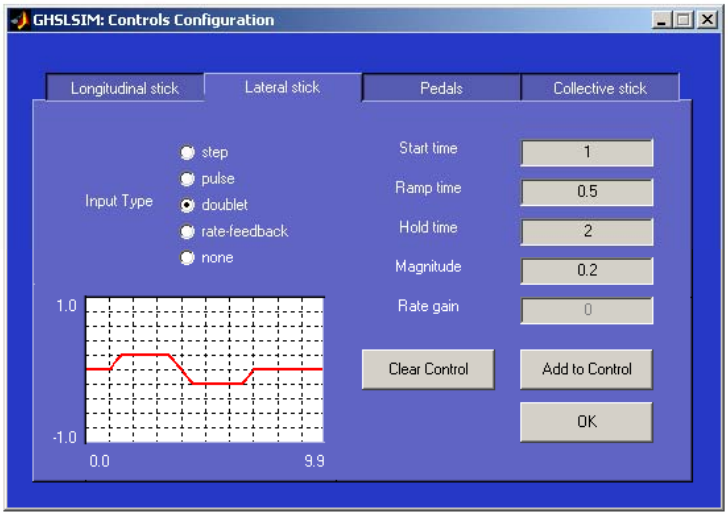

Figure 3.6. Controls Configuration Interface

There are four tabs – one for each control device – including *Longitudinal stick*, *Lateral stick*, *Pedals* and *Collective Stick*. The *Input Types* can be a *step*, *pulse*, *doublet*, *rate-feedback* or *none* and each input has a set of parameters to define its shape. These can be explained more clearly using the diagrams in Figure 3.7. A *step* input is the simplest, and defined by a *Start Time*, *Ramp time* and *Magnitude*. These are, respectively, the time at which the input is started, the time taken to reach maximum amplitude and the value of that maximum amplitude. A *pulse* input has those same three parameters, plus a *Hold time*, which defines the length of time the control is held at its maximum position (magnitude). A *doublet* is essentially two pulse inputs – one positive pulse immediately followed by a negative one - and the *Start Time*, *Ramp time* and *Magnitude* are all the same as previously defined.

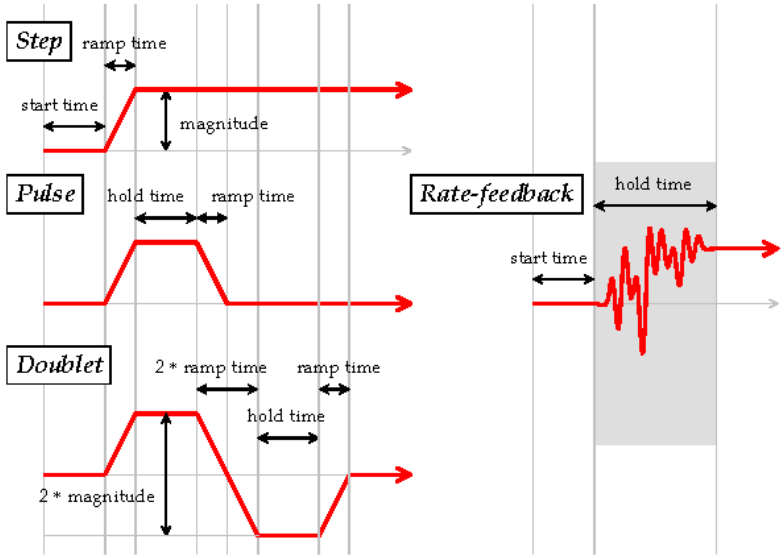

Figure 3.7. Control Input Types

The *rate–feedback* input type is quite different to the former three inputs. It is defined by a *Start time*, a *Hold time* and a *Rate-gain* factor. The *Hold time* basically defines the period during which the *rate-feedback* control is active and the Rate gain defines the magnitude of control feedback employed.

This input is a simple automatic control that augments the commanded controls defined using the above inputs. Equation 2 describes the algorithm used:

$$\begin{aligned} \delta_b &= K_1(q - 0.5 * d\theta) \\ \delta_a &= K_2(p - 0.5 * d\phi) \\ \delta_r &= K_3(r) \\ \delta_c &= K_4(\dot{\&}) \end{aligned} \tag{2}$$

where δ_b denotes the longitudinal stick control, δ_a denotes the lateral stick, δ_r denotes the pedals and δ_c denotes the collective stick. K_1, …, K_4 are the rate-feedback gains for each control. The angular body rates in pitch, roll and yaw are p, q, and r, respectively and the rate of descent is $\dot{\&}$. The pitch and roll angles perturbed from their trim states are denoted by $d\theta$ and $d\phi$. This rate-feedback law has proven sufficient for steady-level flight application, such as initial correction of trim and recovery from a commanded manoeuvre.

Each control input sequence is constructed by repeatedly adding basic input commands with the *Add to Control* button and, if necessary, clearing the entire input sequence with the *Clear Control* button. The basic inputs are superimposed on to the current control sequence and any periods of rate-feedback are marked (shaded). Since the *rate-feedback* essentially replaces any commanded input when it is active, it is unnecessary to have any such input during those periods. However, the *rate-feedback* can be utilised for manoeuvre recovery. So, for example, rather than employing a *pulse* input, one could simply use a *step* input followed by a *rate-feedback* input, as the automatic control will tend to return the system and controls to a state close to their trim state anyway.

Note that the actual commanded input or rate-feedback for each control is offset by its trim position, as determined during the trim stage. This must be kept in mind when generating the control input sequences, since the controls may exceed their limits when adjusted by their trim values and be truncated accordingly.

3.3.4 Trim Initialisation

The last configuration interface sets the initial conditions prior to trim. Shown in Figure 3.8, these variables set the 'initial guess', which the balance and/or trim optimisation stages begin from. Fields include: the Initial Attitude, incorporating *bank*, *pitch* and *azimuth* angles; and the Initial Controls, *Longitudinal stick*, *Lateral stick*, *Pedals* and *Collective stick*. The default for all of these is zero, except the *Collective stick*, which is set to half of its upper limit.

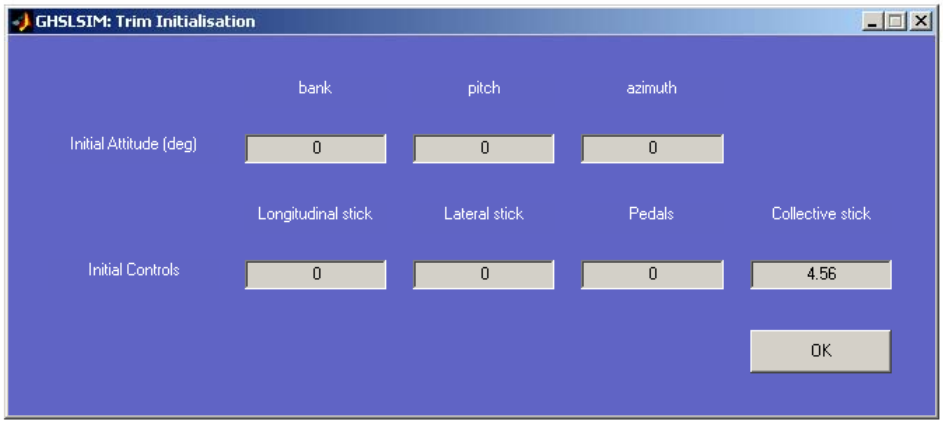

Figure 3.8: Initial Conditions Interface

3.4 Simulation Controls

Once the helicopter and loads have been specified, and the initial conditions set, the helicopter-load system can be *Balanced* and *Trimmed*. Then, assuming that the control inputs have been constructed, the simulation may be *Run*. Strictly speaking, it is not necessary for the system to be balanced, or trimmed before running. However, it is strongly advised that these stages be followed in order, to minimise disturbance during the initial period of simulation. The three stages under the heading of Simulation are described in the following section.

At any point during the balance/trim/simulation stages, the system state and output, if present, can be saved using the file menu. In fact, it is recommended that the user saves to (a different) file regularly, should they wish to repeat a certain stage without having to re-run all preceding stages.

3.4.1 Balance Loads

Because the trim optimisation can be a difficult task for multi-body systems, it is often wise to balance the load(s) beneath the helicopter before trimming the complete configuration. Without doing so, the optimisation routine may get trapped in local minima, or take an excessively long time to reach convergence.

Essentially, this entails holding the helicopter to some fixed position, and attitude while conducting an external force minimisation on each load until they reach equilibrium. The load subsystem is significantly easier to balance, and when converged, provides an initial guess of the system state that is generally much closer than previously set. It is also possible for the user to switch between *inelastic* and *elastic* sling models, thereby simplifying the sling-load model even further for initial balance optimisation.

Figure 3.9 shows the REPLAY display during the load balance stage. The *Start*, *Pause*, and *Stop* buttons are all deactivated during this iterative procedure, and the display frequency slider (*freqn*) and *Tip Trails* checkbox have no effect here. However, the viewpoint azimuth (*azim*) and elevation (*elev*) sliders, and the *Erasemode* checkbox can all be adjusted at any time. Along

with the main three windows (*Display*, *Menu* and *Controls*), a window displaying trace plots of selected variables that are used in the optimisation is shown. For balance, these include the load angles, ϕ, θ, and ψ, as well as the logarithm of the out-of-balance force component, F. This last variable gives an indication of the system's balance during optimisation and should converge to zero. The balance optimisation can be stopped at any time by simply pressing the *STOP* button in the variable-tracking window.

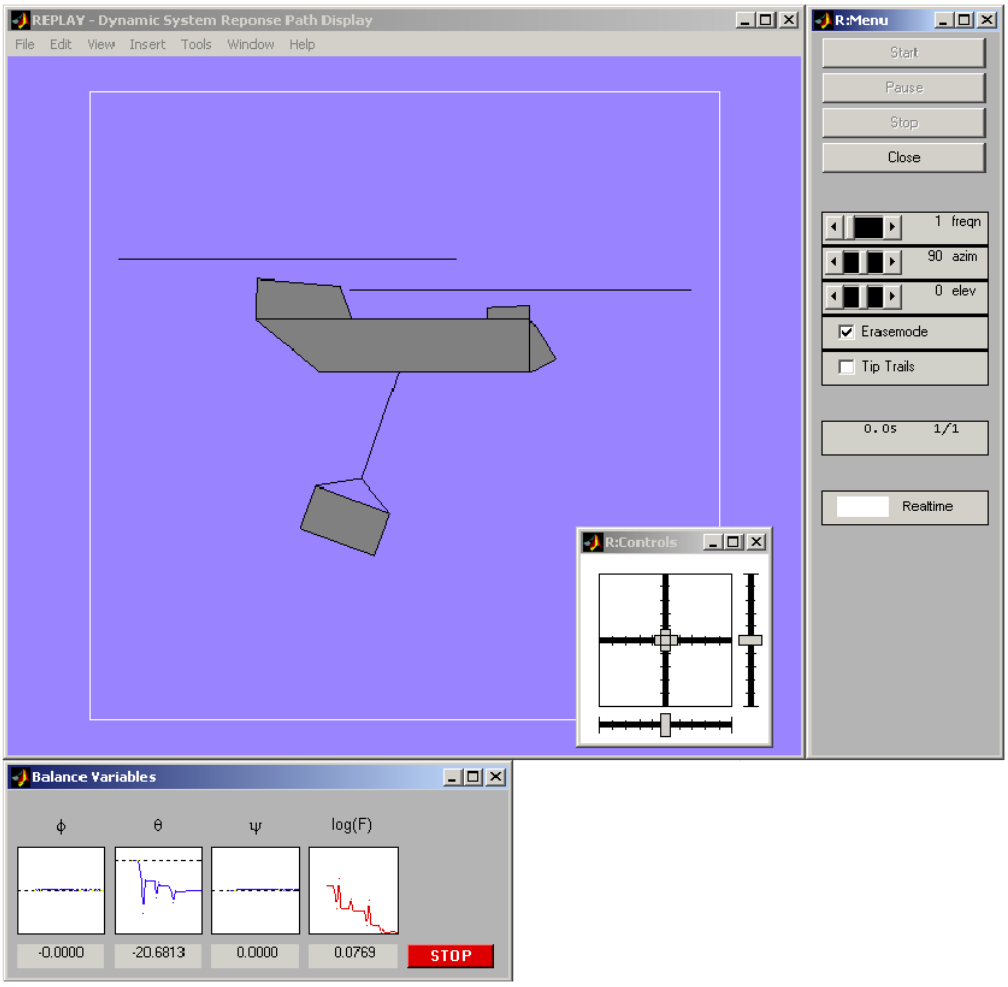

Figure 3.9: Loads Balance Display

If the *Verbose Output* flag has been checked in the program preferences, the optimisation routine writes information to the MATLAB® command window as shown in Figure 3.10. For the balance and trim optimisation, this information includes the iteration number, the cumulative number of function evaluations, the residual, step size and directional derivative. The residual should approach zero, and both the step size and directional derivative should get smaller as the iterations progress. More detail on these indicators is available in the Optimization Toolbox documentation [17]. When the optimisation has finished, the final values for each of the variables shown in the variable tracking window are listed.

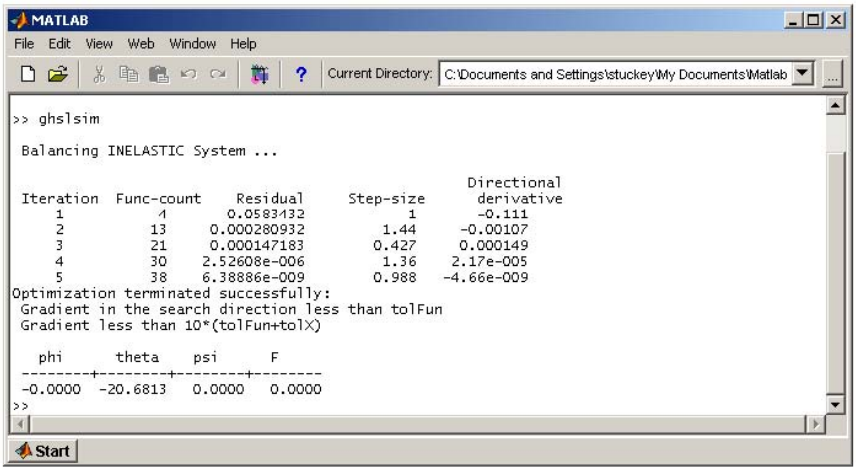

Figure 3.10: MATLAB® Command Window Output

A suggested procedure for balancing the load(s) under the helicopter is as follows:

1. First set the initial conditions for the helicopter, if a good estimate is known, using the Trim Initialisation interface.
2. Set the *Sling model* to *inelastic* in the Main interface, then balance.
3. If the load has balanced adequately, set the *Sling model* to *elastic* and balance again. Otherwise, re-run the balance optimisation, possibly modifying the initial conditions beforehand. This may be necessary if the routine is having trouble converging to a balanced state.

If one does not require an elastic simulation, then the third step is unnecessary. Once balanced, the user may go on to trim the complete system. Since the balance stage operates solely on the load and does not call the ROTORGEN helicopter model, it is less prone to accuracy limitations. However, these may be encountered during the trim stage, as will be explained next.

3.4.2 Trim System

Although not necessary, it is advisable to trim the helicopter-load system prior to simulation to avoid any initial disturbance due to out-of-balance forces. Depending on the fidelity of the simulation required, it is possible to utilise an inelastic sling model, or an elastic one, and the trim routine can handle either. The REPLAY display, shown in Figure 3.11, is much the same as that rendered in the balance stage, apart from the addition of the control input variables. These include, in order, the longitudinal stick δ_b, the lateral stick δ_a, the pedals δ_r, and the collective stick δ_c. The controls window is also activated during this stage, for a clearer indication of the control stick displacements. They are generic slider-type controls and incorporate: the longitudinal and lateral stick, placed vertically and horizontally in the main axes; the pedal controls, placed horizontally below the main axes; and the collective stick, placed vertically to the right of the main axes. The controls are normalised with respect to their hard limits in all of the display axes.

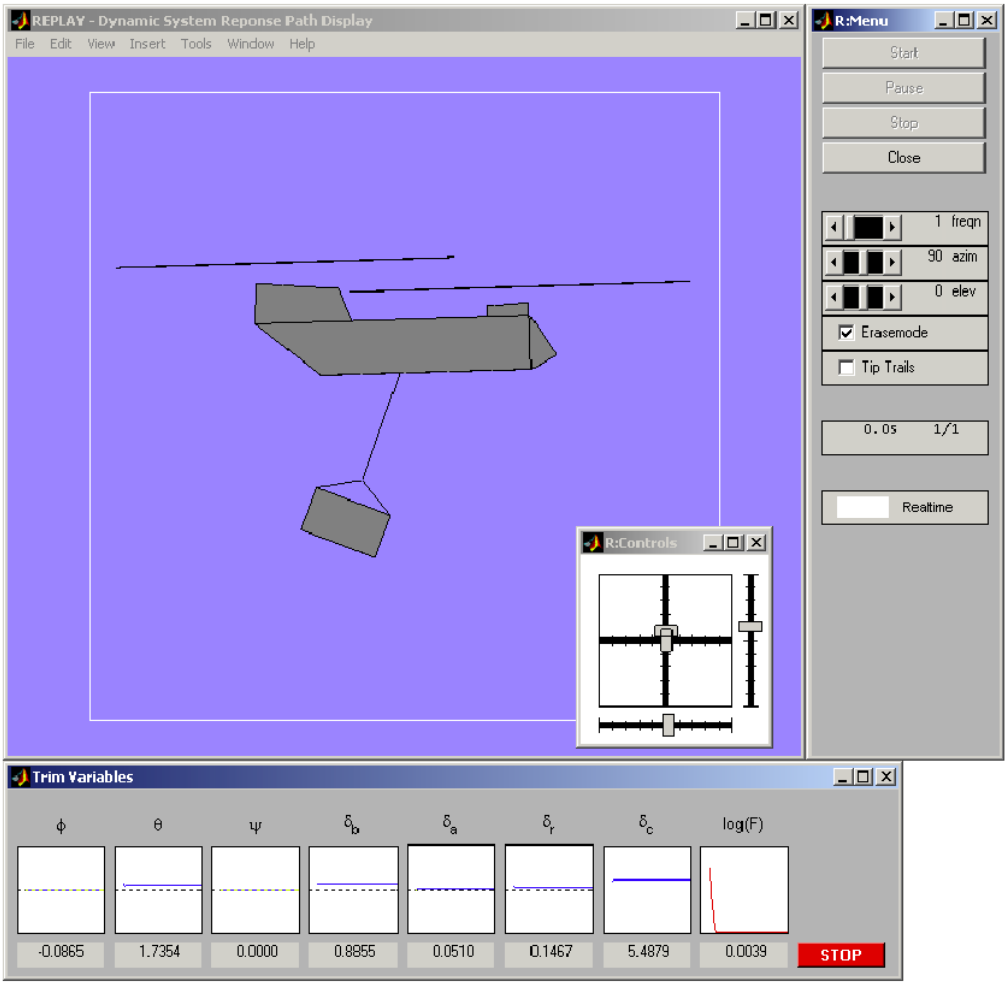

Figure 3.11: System Trim Display

If desired, one can change the viewpoint azimuth and elevation, and the *Erasemode* during trim using the controls in the *Menu*, and as with the balance stage, it is possible to stop the trim optimisation at any time using the button displayed.

The suggested procedure for trimming the helicopter-load system is:

1. First set the initial conditions for the helicopter, and balance the load under it.
2. Trim the system using either in inelastic sling model, or an elastic one as required. It is not necessary to do both.
3. If the system has not reached trim, re-run the trim optimisation. Check that the load is balanced. If the system still does not trim, try a different set of initial conditions. Failing this, it may be necessary to fly the helicopter and load into an approximate trim state through simulation. See Section 3.7.2 for a further discussion.

Note that for some configurations, it may be difficult to find a satisfactory trim point. For example, a lightweight load flown under a helicopter at high speed will tend to be very unstable. Moreover, it may not be physically possible to attain trim at all, such as the case when a load is too heavy for the helicopter to lift. The user should be aware of the limitations specific to each configuration before attempting trim or simulation using GHSLSIM.

Unfortunately, there are also accuracy limitations in the ROTORGEN (MEX) helicopter model. MATLAB®, and the models generated within the environment, such as the loads, are all intrinsically double-precision, whereas the ROTORGEN model is much less accurate. As a consequence, truncation error may cause the trim optimisation to get 'stuck' in a discrete state, or it may not converge to a satisfactory solution. Often however, the trim optimisation will reach a satisfactory state within the maximum number of iterations, even though the function may suggest the optimisation be re-run (possibly increasing the maximum number of function evaluations). If it appears to have done so, according to the residual error, then it can be considered correctly trimmed.

3.4.3 Run Simulation

The REPLAY window appears again for the simulation stage, shown in Figure 3.12. In the variable tracking window, the helicopter inertial velocities ($\dot{x}$, $\dot{y}$, $\dot{z}$) are plotted, as well as the helicopter angles and control inputs. The REPLAY function is still called iteratively, so none of the main menu commands function, however, a counter window provides an indication of the simulation progress and the simulation can be stopped at any time as with the trim stage. In addition, if the *Verbose Output* flag has been checked in the program preferences, the integration routine will output the time, the absolute helicopter velocity, the relative load velocity and an estimate of the integration error in the MATLAB® command window.

Typically, the system is too complex to run in real-time and therefore, simulations with a small timestep, or a high order integration algorithm will take longer to complete. However, if the simulation has diverged due to truncation error, it may be necessary to increase the order, or reduce the timestep.

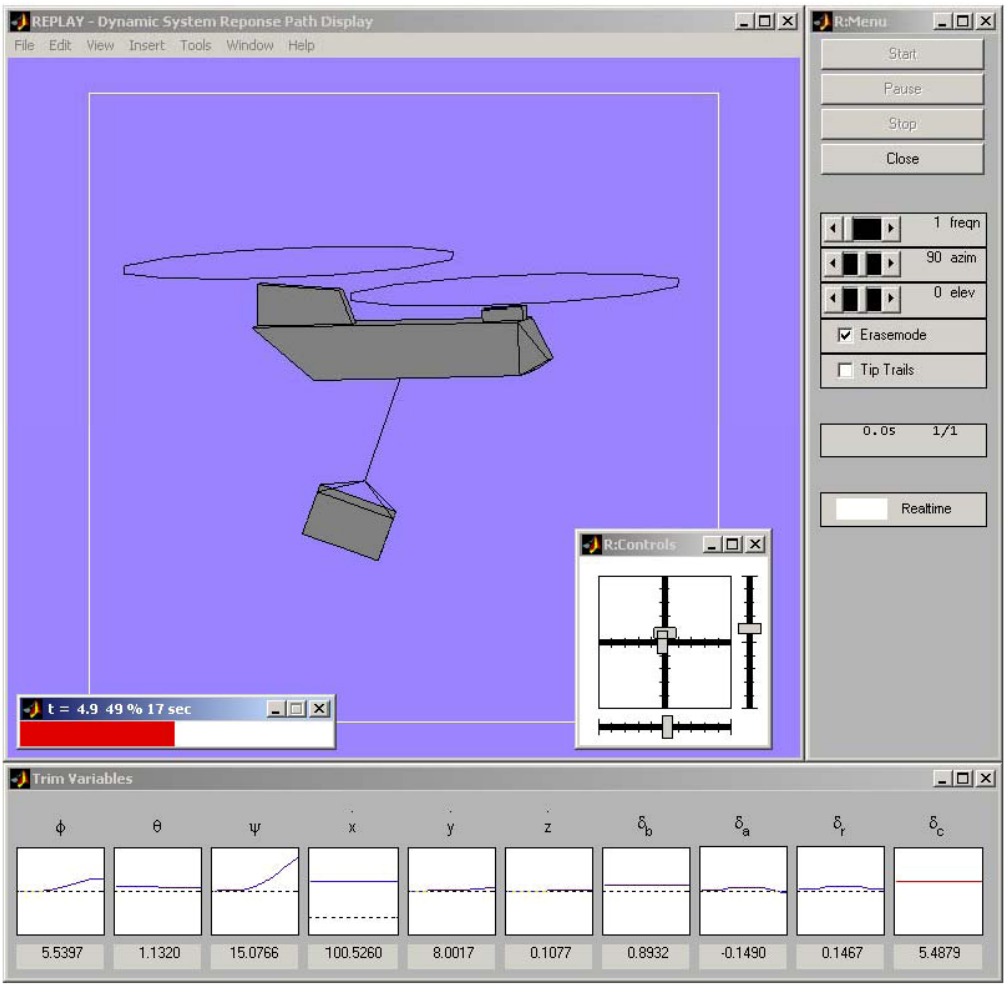

Figure 3.12: Simulation Run Display

3.5 Output Controls

The output controls are intended to be used following simulation, and include buttons to *Replay Simulation*, *Plot Time-Histories*, and *Generate VRML* code. Both the simulation replay and VRML model provide a real-time three-dimensional graphical display of the simulation, whereas the time-history plots are intended for analysis of the output data.

3.5.1 Simulation Replay

This facility opens the same REPLAY windows as for the simulation stages; however, this time it is called with the full position, orientation and control data sets, rather than just the data at any particular step in an iterative fashion. Executing the REPLAY function in this way allows full control of the *Start*, *Pause*, and *Stop* commands in the menu, as well as several options not previously available. An example of the display is shown in Figure 3.13.

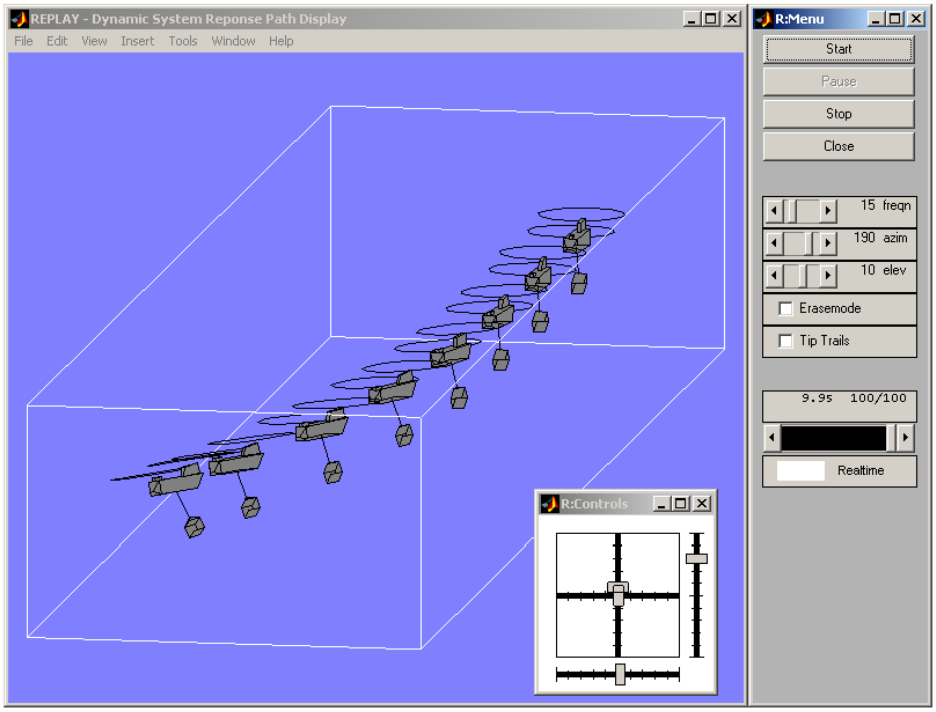

Figure 3.13: Simulation Replay

Freqn denotes the integer display frequency for the simulation, that is, the number of frames skipped before each display step. For example, a *freqn* of 1 will display every frame, and a *freqn* of 10 will display every 10th. The viewpoint can also be moved about 360° in each axis using the *azim* and *elev* sliders. If the *Erase mode* option is checked (the default), each frame will be erased before rendering the next. If it is unchecked, a trail or history of frames, such as that shown in the figure will be displayed. Note that these frames aren't permanent objects, and once the viewpoint changes the trail will disappear. Another option to help with the visualisation is to display *Tip Trails* during the simulation replay. The *Tip Trails* are simply coloured lines that follow the port and starboard extremities of the helicopter, much like long streamers. Below the controls and options is a frame displaying the current time and timestep. The associated slider can be moved to any point within the simulation range and the helicopter-load model will be set to the corresponding position. The last box in the menu window is the *Realtime* indicator. During replay, regardless of the display frequency, this changes colour, depending on whether the simulation is being replayed in real time: green indicates real time and red indicates a refresh rate slower than real time.

3.5.2 Time-History Plots

The time-history plots give a qualitative overview of the simulation run for the helicopter and load(s). The variables that are plotted depends on the *Simulation axes* defined in the simulation configuration, as well as the control inputs and the number of loads. An example of one set of plots is shown in Figure 3.14, for a single-load configuration.

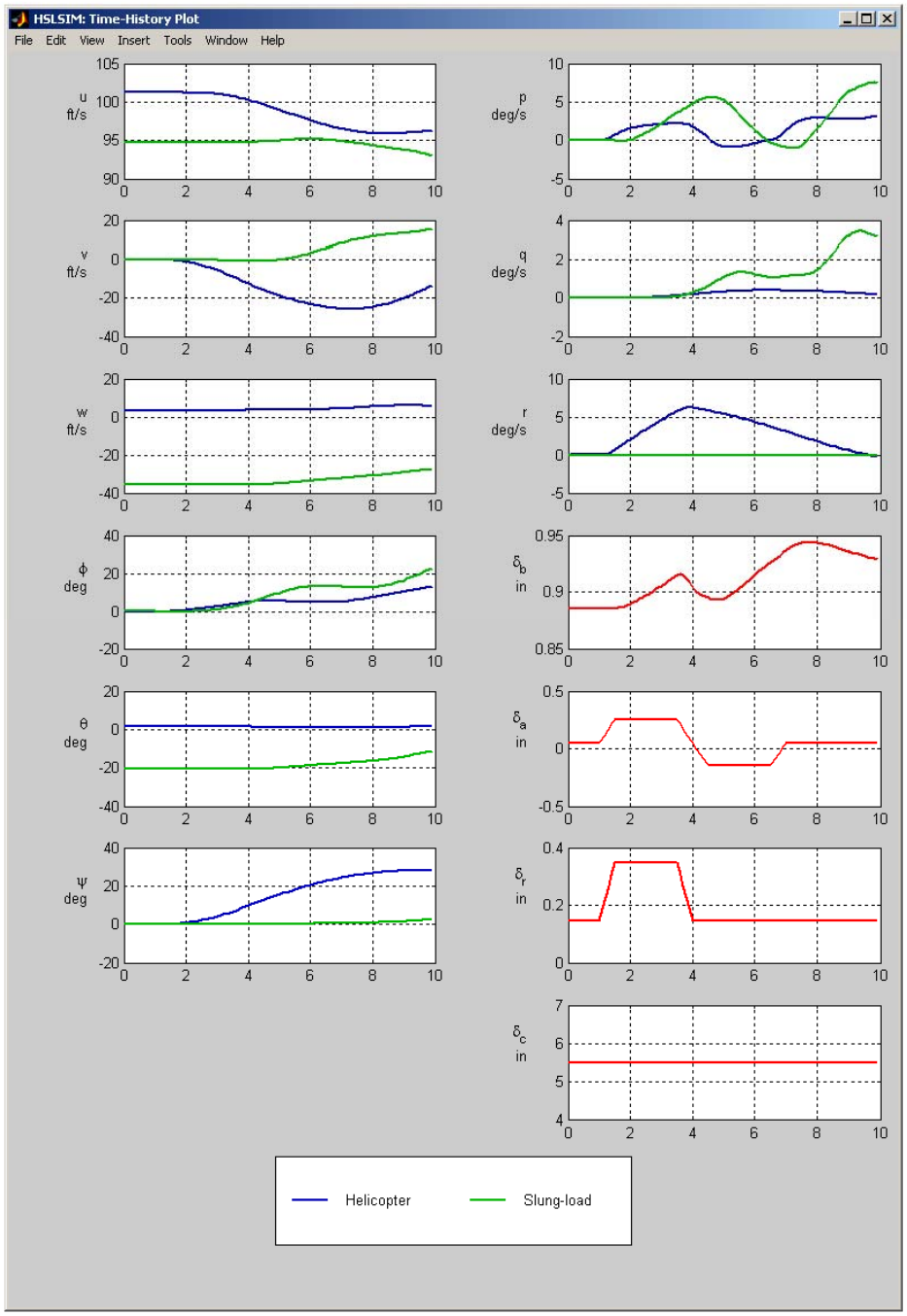

Figure 3.14: Time-History Plots

Here, the body-axes velocities (u, v, w) and the body-axes angular rates (p, q, r) are included, as well as the Euler angles (ϕ, θ, ψ) and the four control inputs (δ_b, δ_a, δ_r, δ_c).

3.5.3 VRML Generation

The Virtual Reality Modelling Language (VRML) is an ideal language for the visualisation of dynamic flight simulations [18]. Using the VRML Generation command, the user can export the simulation in VRML format, which can then be viewed and replayed in a VRML browser. Figure 3.15 is a screen capture of the Internet Explorer web browser with the Cortona VRML plug-in installed.

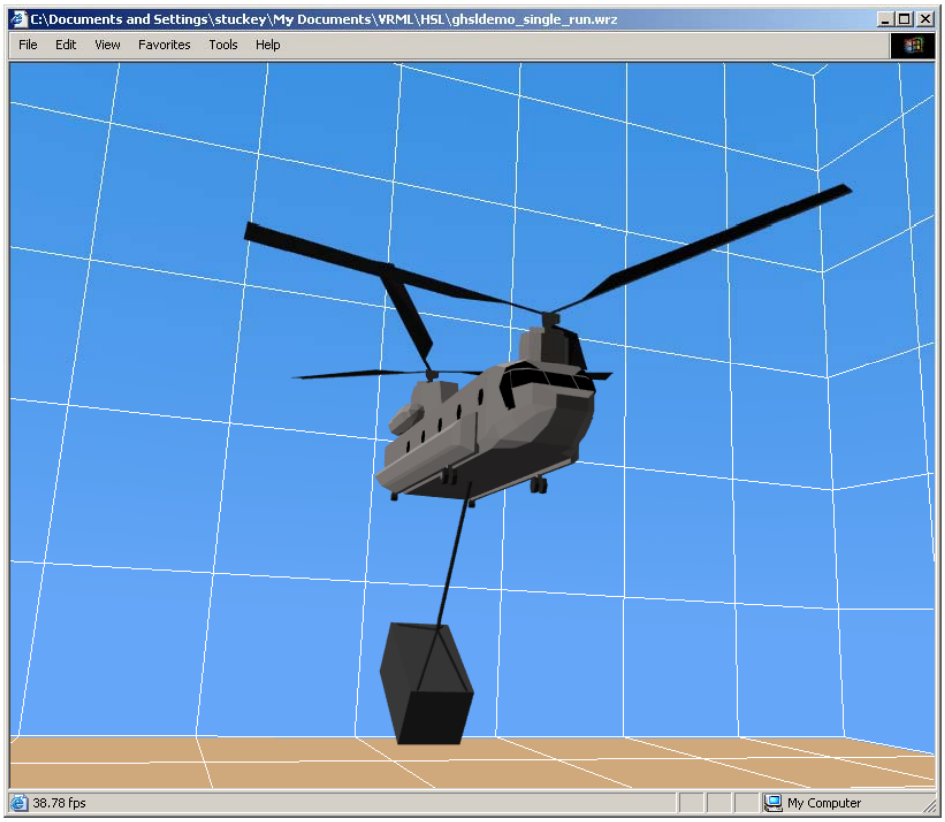

Figure 3.15: VRML Simulation Replay

It is important to make sure the *VRML Directory* is set correctly in the program preferences, that is, the directory in which all the VRML helicopter and load models reside. Otherwise, they will not appear in the VRML simulation.

To run the simulation replay, the user simply clicks on either the helicopter or load. It is not possible to pause or stop the simulation midway through the run, and to go back to the first frame without restarting, the user must reload the model (using the browser's reload command).

There are several viewpoints to choose from, including *User*, *Side*, *Front*, *Top*, and *Behind*. These are all static, in the sense that their inertial position, relative to the helicopter, is fixed throughout the simulation, whilst the view angle remains constant. The dynamic viewpoints

include *Fixed-Inertial*, *Fixed-Zoom-Pan*, *Fixed-Aircraft* and *Fixed-Down*. Unlike the first set, the view angle for these changes during the simulation. The *Fixed-Inertial* viewpoint represents one whose position is fixed in space, such as an observer on the ground. *Fixed-Zoom-Pan* is the same, except the focal length changes with time such that the helicopter-load configuration occupies the full view throughout. The *Fixed-Aircraft* and *Fixed-Down* viewpoints have fixed offsets from the helicopter, which remain constant and their view angles change with any variation in the helicopter angle. The former is located at a position and angle similar to the User viewpoint, while the latter is located just below the middle helicopter attachment point, looking down at the load. As well as the listed viewpoints, it is possible to navigate the viewpoint to any position and angle in space using the VRML browser's controls. When doing so, it is important to remember that the variation of the viewpoint position and/or angle is not altered, just the offset. So, for example, changing the initial position of the *User* viewpoint will follow the helicopter as expected, but changing the initial position of the *Fixed-Inertial* viewpoint may result in a strange replay that does not follow the helicopter at all.

In addition to the helicopter, load and cables, a load displacement patch is superimposed on the VRML model to aid with visualisation of the load's swinging motion. This patch extends from the helicopter attachment point to the load centre-of-gravity and indicates the amount of relative displacement seen by the load from its trim position.

3.6 Example: 3000 L Water Tank

An example is given here for the CH-47D and a 3000 L water tank, suspended by four slings, as shown in Figure 3.16. This is a simple load and can be adequately represented by a box-shaped container. The centre of gravity is assumed to remain at the centre of the tank at all times.

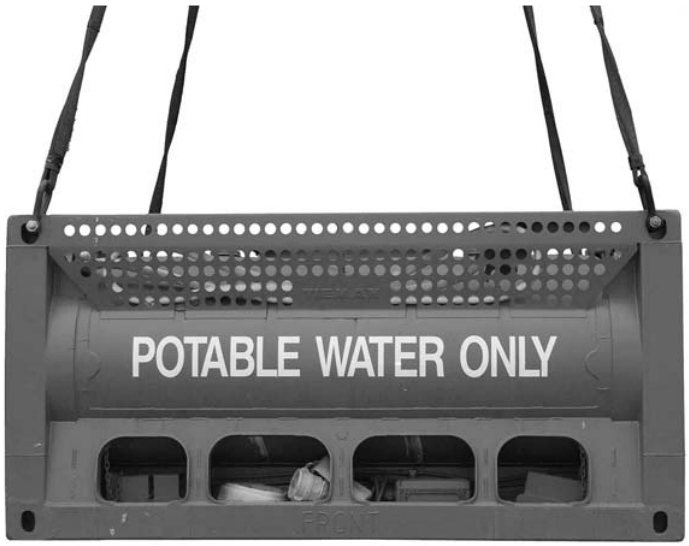

Figure 3.16: 3000 L Water Tank

The control inputs specified include a rate-feedback on the longitudinal stick to keep the nose level throughout the manoeuvre, as well as a doublet on the lateral stick and a pulse on the pedals to excite the load. For this case, they are not intended to accurately represent any real-world inputs that might typically be employed, but specifically designed to excite the load under the helicopter.

Below is a list of the steps to configure, trim and simulate the helicopter load system. They represent just one approach, especially for trim, that can be used to achieve the same outcome. For example, another scheme to obtain trim would be to first balance the *inelastic* configuration, then trim the *inelastic* configuration, and then trim the *elastic* configuration.

The steps are as follows:

1. Start the program, by typing at the command prompt:

   >> ghslsim

2. First, check the program preferences. Select File → Preferences... from the main menu. Confirm that the *Web browser*, *VRML Directory* and *GZIP Executable* fields are correct and if not, update them accordingly. Check all of the checkboxes for a detailed output and visualisation.

3. Next, in the main interface, set the following fields:

Simulation time (s)	10
Integration timestep (s)	0.1
Number of steps	101
Trim airspeed (ft/s): Forward	100
Trim altitude (ft)	100
Simulation axes	◉ combined
Angular representation	◉ Euler
Simulation type	◉ nonlinear
Sling model	◉ inelastic
Trim routine	◉ lsqnonlin
Simulation routine	◉ ode45

   This will create a 10 second simulation with 101 timesteps. The forward speed is approximately 60 kn and the altitude 100 ft. All options are set to their defaults, except for the *Sling model*, which is set to *inelastic* for the balance stage.

4. Press the Configure helicopter button. In the Helicopter Configuration window that pops up, type in the following options:

Helicopter type	◉ CH-47
Simulation model	◉ MEX

   For this run, leave the *Stability & Control Augmentation System (SCAS)* checkbox unchecked.

Weight (lb)	46000
Inertia (slug.ft^2): Ixx; Iyy; Izz	43200; 250000; 234000

   Press the OK button to return to the main interface. These are the figures for a fully loaded Chinook CH-47D, with the SCAS switched off.

5. Press the Configure loads button. In the Helicopter Configuration window that pops up, type in the following options and fields:

Load type	◉ box
Load weight (lb)	3330
Load inertia (slug.ft^2): Ixx; Iyy; Izz	1000; 4000; 4000
Size (ft): Length; Width; Height	8.00; 6.46; 3.84
Sling configuration	◉ Multiple point
Cable length (ft)	19.5
Stiffness (lb_f/ft)	20000
Damping (lb_f/ft/s)	500
Attachments (ft): x-offset; y-offset; z-offset	4.00; 3.23; –1.92 –4.00; 3.23; –1.92 –4.00; –3.23; –1.92 4.00; –3.23; –1.92

Leave the *Cables slack (zero force) under contraction* checkbox unchecked. This represents an empty 3000 L water tank, supported by four slings attached to each upper corner of the load. Since there is only one load with a multiple-point configuration, the slings will automatically be located to the helicopter middle cargo hook. At this stage, there is no attempt to implement the correct centre-of-gravity position, and consequently, it is placed at the centre of the load. Press the OK button to return to the main interface again.

6. Press the Configure controls button. In the Controls Configuration window that appears, make sure the *Longitudinal stick* tab is active and type in the following options:

Input type	◉ rate-feedback
Start time (s)	0
Hold time (s)	10
Rate gain	40

Now press the Add to Control button to add the specified input to the longitudinal control. The entire control graph should fill with grey, indicating the region of control feedback. Next, press the *Lateral stick* tab to activate that control and type in the following:

Input type	◉ doublet
Start time (s)	1
Ramp time (s)	0.5
Hold time (s)	1
Magnitude (in)	0.2

Press the Add to Control button to add the specified input to the lateral control. A doublet input with those properties will be drawn to the control graph. Then within the same frame (*Lateral stick*), type in the following:

Input type	◉ pulse
Start time (s)	4.5
Ramp time (s)	0.5
Hold time (s)	0.5
Magnitude (in)	–0.2

Again, press the Add to Control button to superimpose the specified input to the lateral control. Adding this pulse effectively lengthens the second, negative component of the doublet-type control sequence. Next, press the *Pedals* tab to activate that control and type in the following:

Input type	◉ pulse
Start time (s)	1
Ramp time (s)	0.5
Hold time (s)	2
Magnitude (in)	0.1

Press the Add to Control button to add the specified input to the pedal controls. A pulse input will be drawn to the graph. Next, press the *Collective stick* tab to activate that control and type in the following:

Input type	◉ none

Press the Add to Control button to add the null input to the collective controls. Lastly, press the OK button to return to the main interface.

7. Press the Initial Conditions button. In the Trim Initialisation window that appears, type in the following options:

Initial Attitude (deg): bank; pitch; azimuth	0; 5; 0
Initial Controls (in): Longitudinal; Lateral;	0; 0
Pedals; Collective	0; 4.56

The helicopter should have a slight, positive pitch angle at trim, hence the initial guess of 5°. Each of the controls is typically set to its central (range-average) positions. For the Collective stick, which ranges from 0″ to 9.12″ for the CH-47D, this is 4.56″, and for the rest it is zero. When finished, press the OK button to return to the main interface.

8. Save the simulation configuration data via the main menu: File → Save As. It is highly recommended that the user saves at this point, in order to avoid having to re-enter all of the above data should anything go wrong! This will save all the configuration data and the initial trim state guess. Now, if the configuration needs to be altered, or any subsequent balance, trim or simulation re-run, it is simply a matter of loading the saved file via the same menu (File → Open), and resuming from there.

9. First step of the simulation stage: Balance the loads by pressing the BALANCE LOADS button. This first simple inelastic configuration should balance within a few iterations. The load should have a pitch angle of approximately –5°, that is, swung aft by 5° due to the bluff-body drag in effect at 100 ft/s.

10. Now that a better idea of the initial state has been determined, one can proceed to the elastic configuration with confidence. Set the *Sling model* to *elastic* with the radio button on the main interface and balance again (BALANCE LOADS). The cables will stretch slightly to counteract the force exerted by the load weight and drag. Consequently, the load position and angle will change, but not by much. The pitch angle, for example, will still be approximately –5°.

11. Proceed directly to trim, without making any further alterations to the configuration. Press the TRIM button. This time, the helicopter's attitude and controls will change, as well as the load's position and angle. Since the algorithm is now optimising the full 12 degrees-of-freedom, it will take slightly longer to converge. For this configuration, the nonlinear least squares trim routine will take around 50 iterations before the tolerance criterion is met. The resulting trim values displayed for the helicopter are:

Table 3.2: Helicopter Trim State with 3000 L Water Tank Load

phi	theta	psi	del_b	del_a	del_r	del_c
-0.10	2.95	0.00	1.00	0.06	0.16	5.37

Save the configuration data via the main menu as before. If the same file is chosen (or saved directly, using File → Save), this will update the Trim Initialisation variables to their new values. Saving at this point can be considered optional, although it is recommended if multiple simulation runs are to be executed from the same trim state. For example, if the user is interested in testing the effect of different control inputs on the same configuration.

12. Run the simulation with the RUN button. For the 101 time steps, this takes around 30 seconds on an Intel Pentium-4 1.9 GHz processor machine. If the *Replay Window* option was checked, the user will see the helicopter and load being simulated as the integration routine progresses. When the simulation has completed, a new Replay window will appear with the full simulation ready for replay.
13. Save the simulation configuration *and* output data via the main menu: File → Save As. Once again, it is highly recommended that the user saves at this point.
14. If the *Replay Window* option was not checked, or the Replay window was closed, it is still possible to replay the simulation using the Replay Simulation button under the Output controls. Close the windows using the button on the Replay menu when finished.
15. Plot the time-histories using the Plot Time-Histories button.
16. Generate VRML model and replay with the Generate VRML button. The resulting VRML file will be automatically saved to the *VRML Directory* specified in the preferences. Exit the program.

The configuration can now be re-trimmed, or the simulation re-run from the files saved earlier. It is also possible to replay the simulation again, or plot the time-histories from any complete run that was saved.

3.7 Advanced Topics

The Graphical User Interface (GUI) is sufficient for most purposes, and provides a convenient tool for slung load simulation and analysis that requires very little knowledge of the underlying system. Indeed, the GUI is just a higher-level simplified interface to the full HSLSIM package. Many options are hidden and various assumptions are made regarding the configuration details within the program. However, it is possible to access the lower-level scripts and functions of the package, and once the user has gained some experience in using it, they may wish to write their own scripts, or tailor the code to meet their requirements.

3.7.1 Automation & Multiple Simulation Runs

The following section details an example for the automated execution of a suite of simulation runs. In this example, the configuration from the previous section is used, that is, a CH-47D helicopter and 3000 L water tank suspended by four slings. The aim of the experiment is to assess the effect of airspeed and load weight on the stability of the load during lateral doublet manoeuvres. To this end, the maximum load deviation will be recorded for a suite of simulation runs over a range of trim airspeeds and load masses. Refer to Appendix A.2.1 for a full listing of the code, **`ch47b3kltsim.m`**.

Variables that can be safely modified by the user are all defined at the top of the file in the configuration section. These include **`B_title`**, **`B_mdir`**, **`B_V`**, **`B_m`**, **`B_T`**, **`Prefs`**, **`Config`** and **`wrlfiles`**. The variables **`B_title`** and **`B_mdir`** specify the filename prefix and directory used when the data from each run is saved to disk. It is not compulsory to incorporate the title into the directory name, but has been done for clearer organisation. The three vectors, **`B_m`**, **`B_V`**, and **`B_T`** detail the flight-points in terms of load mass, airspeed, and control-input magnitude, respectively, at which the configuration is to be trimmed and simulated. **`B_m`**, consists of 10 points spaced uniformly between 3330 lb and 9943 lb, **`B_V`** consists of 6 points between 60 kt and 120 kt, and **`B_T`** consists of just three points at 0.0, 1.0 and 2.0. Consequently, several of the resulting special variables employed to store results, such as the maximum load deviation **`B_dA`**, will have three dimensions and be of size **`B_mN`** $\times$ **`B_VN`** $\times$ **`B_TN`**, corresponding to the length of each vector ($10 \times 6 \times 3$). The **`Prefs`** struct contains general program preferences, as specified from within the GUI. These include the VRML Directory, the location of the GZIP Executable if present, and several display flags. The Web Browser field is not used here, since the browser is not launched automatically at any stage. The **`Config`** struct contains data for the simulation, helicopter, load, controls and trim initialisation. Most of these fields should be self-explanatory, with a few exceptions. Firstly, note that the horizontal trim velocity is initially equal to **`B_V(1)`** in ft/s, hence the conversion factor. The load weight is equal to **`B_m(1)`**, and the load inertia (in x, y, and z) is approximated by a simple linear function of **`B_m(1)`**. Likewise, the magnitude of the lateral stick and pedal inputs (2nd and 3rd elements of **`ctrldata`**) are functions of **`B_T(1)`**. These are all updated during each iteration of the main loop. Note also that the number and order of the **`loaddata`** attachment offsets are important and specific to the type of sling configuration used. For a single attachment, there is only one set of (x, y, z) locations. For a bifilar configuration, there are two, ordered from the forward attachment to the aft. For both multiple and tandem configurations, there are four, ordered: forward-starboard, aft- starboard, aft-port, and then forward-port (i.e., clockwise ordering). Both the helicopter and load inertias are represented by the traditional 3×3 matrices [19].

Lastly, pay careful attention to the braces used for each field – some are matrices or scalars (with square brackets), and some are cells (with curly brackets). The 2nd and 3rd elements of **wrlfiles** define the location of the helicopter and load models to be referenced in the output VRML file. The 1st element is left empty in this section, but is filled later with an automatically generated filename.

Other variables that are created include the main index matrix **B_I**, which specifies the order of the trim-simulation runs. Since the initial trim state guess for each iteration is based on the previous trim state determined (through optimisation), this order is arranged to minimise the difference between consecutive trim points. Hence, the main loop first ascends through the load mass sequence at the minimum velocity, then descends through the load mass sequence at the next highest velocity, then ascends through the load mass sequence again for the next highest velocity and so on. This zig-zag path over the mass-velocity space is just one scheme that can be employed to cover the range of flight-points. The main aim of any such scheme is to start from the best possible guess of trim at each flight-point and therefore minimise the occurrence of unconverged cases and/or errors. The matrix **B_dA** holds the maximum load deviation for each flight-point. **B__** holds a flag for each flight-point, where a value of 0 indicates that the configuration is untrimmed, 1 indicates that it is trimmed and 2 indicates that it has been (trimmed and) simulated. The matrix **B_u0** holds the trim configuration velocities, **B_r0** holds the trim positions and orientations, **B_xt** holds the trim states and **B_d0** holds the trim controls for each flight-point.

The main functions, **hsltrim** and **hslsim** actually use a set of different variables, as opposed to **Config** and **Prefs**, in which the options and configuration details are kept. These are the global variables **HDAT_**, **LDAT_**, **CDAT_**, **opt_**, and **pref_** . The first three hold the equivalent helicopter, load and cable data and the next two hold the program options and preferences. For more detail, see the script **ghs_init**, which converts the data in **Config** and **Prefs** to the global variables used.

Upon execution, the matrix **B__** is checked to see if the script has been run before and if not, it creates all of the above variables and proceeds to the main loop. If it has, this first step is skipped, as well as any flight points encountered during each cycle of the loop that have been successfully trimmed and simulated. If any particular flight-point has been trimmed, but not simulated, the remaining simulation cases will be run from the corresponding trim state. In other words, the program attempts to trim at each flight point, and then, if successful, runs each of the simulation cases for that point. For each flight-point following the first, the trim state used to initialise the trim optimisation is that determined from the previous flight-point. If the system cannot be trimmed to within tolerance at any particular flight-point, the trim state used is that from two iterations ago. If the system still cannot be trimmed, the trim state from three iterations ago is used, and so on until the list of all previous states has been exhausted. This effectively reduces the likelihood of the optimisation being unable to converge because of a poor initial trim state guess.

At every step of the main loop, the load mass and inertia, and the trim velocity are changed to their appropriate values as given by **B_m** and **B_V**. As mentioned above, the inertia is approximated by a simple linear function of **B_m**. The vertical location of the water tank's centre-of-gravity is also approximated using a simple function based on the water level for any given weight m_j. Since the water level is 2.0 in below centre for both empty (m_1 = 3330 lb) and full (m_N = 9943 lb), and the height of the tank is 46.0 in, the cg offset can be expressed:

$$z_{cg} = 25.0 - \frac{1}{m_j}\left(23.0 * m_1 + \frac{23.0 * (m_j - m_1)^2}{(m_N - m_1)}\right) \qquad (3)$$

After setting the load mass and inertia, and the trim velocity for the current flight-point, the balance/trim stage is commenced. This is done in three steps: First, balance the inelastic load configuration; then balance the elastic load configuration; and lastly, trim the entire helicopter-load system. For each balance step, the following commands are executed:

1. Sling model is set to elastic or inelastic
2. Global variables are initialised with **ghs_init**
3. Load vertical cg position is calculated and used in the geometric sub-struct **LDAT_.S**
4. Initial load position is set directly beneath the fixed helicopter
5. Replay window is created if flagged in the program preferences
6. Load is balanced beneath the fixed helicopter via **hslload**
7. Load positional trim data is updated

The trim step is much simpler, entailing the following commands:

1. Global variables are initialised with **ghs_init**
2. Replay window is created if flagged
3. Trim time-control matrix **TD0** is set to its initial value
4. Helicopter load system is trimmed via **hsltrim**
5. Trim control vector **d0** is updated

The sum of squares of the residual external force is then checked against a tolerance (0.1 in this example), and if unsuccessful, the trim index is decremented, the corresponding element of **B__** is set to 0 and program control returns to the top of the loop for another attempt at trim starting at the previous flight-point trim state. If the check is successful, the matrices **B_u0**, **B_r0**, **B_d0** and **B_xt** are updated, the corresponding element of **B__** is set to 1 and program control continues on to the simulation stage.

Upon entering the simulation stage, the matrix **B__** is examined to find which simulations corresponding to the range of control input magnitudes, have yet to be run. Then, another

loop is started in which those simulations are completed. In this stage, the following commands are executed:

1. Global variables are initialised with **ghs_init**
2. Replay window is created if flagged in the program preferences
3. Trim time-control matrix **TD0** is set to it's initial value
4. Configuration velocity vector **u**, the position & orientation vector **r** and the full state vector **x** are set to their initial values
5. Time-control input matrix **TD** is generated for the specified time sequence and offset according to its trim value
6. Simulation is run via **hslsim**
7. Data struct is created with the simulation output, including time **t**, state **X**, controls **C**, modified controls **CC**, body-axes velocities **Va**, body-axes accelerations **Vadot**, cable angles **Acj**, load deviation **Ajj**, and cable force **Fc**
8. Save the **Config** and **Data** structs to file
9. Update the maximum load deviation matrix **B_dA**

The last step of the simulation stage involves creating the VRML file from the simulation data output. In order to accomplish this, first, the load's vertical cg location must be reset back to centre, and the load position time-history needs to be adjusted accordingly (in effect, removing the offset, z_{cg}). Then, the user and fixed-inertial viewpoint offsets are defined and the VRML file generated via the function **hslvrml**. If the executable GZIP exists, the file is compressed from its native ASCII format to binary. Lastly, the corresponding element in the special matrix **B__** is set to 2 to indicate a successfully simulated flight-point and both loops (balance/trim and simulation) are closed. The progress and/or total execution time is displayed and all variables used in the script are saved to file. These include all of the special variables **B_*** used in the script to store data from each flight-point.

If for any flight-point, the program has too much difficulty in attaining trim (and the user is certain that trim is possible for that particular configuration and flight-point), a different approach may be necessary. One method is to vary the initial guess of the trim state variables, **r0** and **d0** to try and eliminate any optimisation problems with local minima. This can be done, either from within the GUI, or by executing commands from the workplace itself. In the latter case, should trim be attained, it is imperative that the special matrix **B__** be updated to indicate a successfully trimmed flight-point. Another alternative is to modify trim procedure of the main script to account for difficult cases or even just that particular flight-point. Perhaps the simplest solution however, is to extend the time frame by some period preceding the commanded manoeuvre in which the helicopter and load are stabilised using rate-feedback. This was implemented for the trim and simulation of the CH-47D and Merlo Rough Terrain Forklift, in the script **ch47bmrtfsim.m**. The simulation time was increased to 15 seconds, the first 5 seconds of which, the helicopter was controlled by rate-feedback on the stick and pedals. Consequently, the helicopter is still able to stabilise from an initial state that is not adequately trimmed at several flight-points.

There are various ways in which the data from a simulation experiment can be analysed. The script **hslmap** is one example, which plots colour contour maps of the maximum load deviation stored in the matrix **B_dA**. The script can be run immediately after a set of trim/simulation runs, or after loading the final file (above) containing all relevant variables from such an experiment. **hslmap** creates one plot per control-input magnitude specified in the vector **B_T** as shown in Figure 3.17, Figure 3.18, and Figure 3.19. Each plots illustrates the variation of maximum load deviation with, in this example, load mass and airspeed. Contour lines are also added to the plots for more clarity. For the CH-47D helicopter and 3000 L Water Tank load, it can be seen that the maximum load deviation generally decreases with increasing airspeed and load mass, and increases with an increase in the control input magnitude. There do not appear to be any anomalies in the contours.

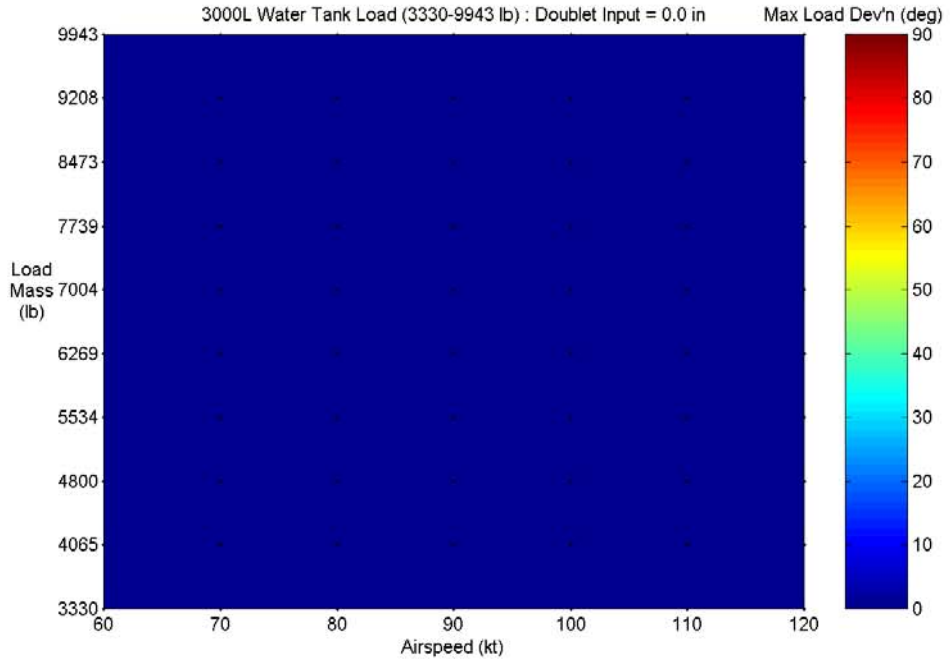

Figure 3.17: Max Load Deviation Contour Plot: CH-47D + 3000 L Water Tank, Lateral Doublet Input Magnitude = 0.0 in

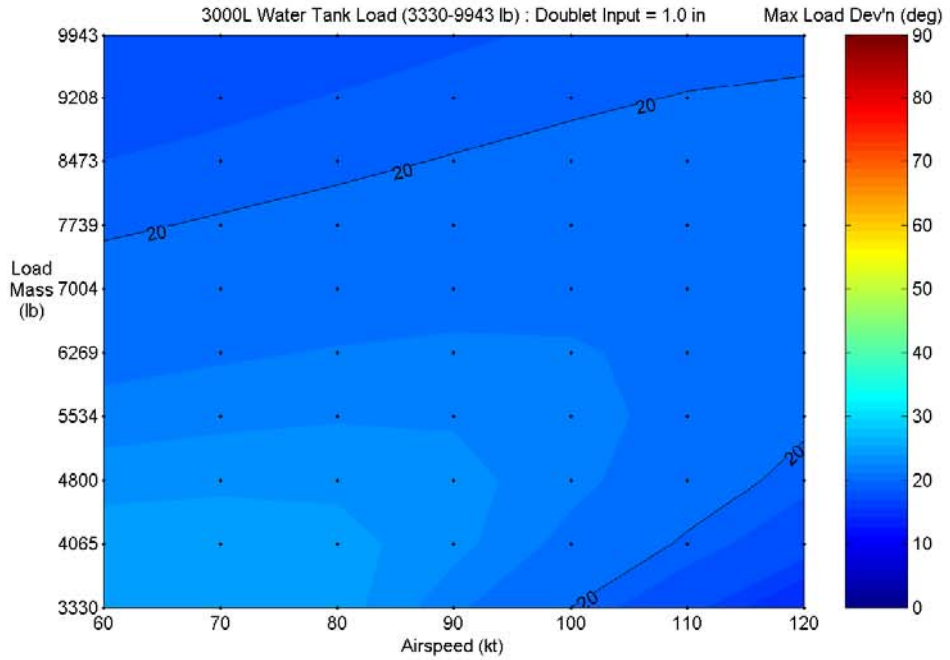

Figure 3.18: Max Load Deviation Contour Plot: CH-47D + 3000 L Water Tank, Lateral Doublet Input Magnitude = 1.0 in

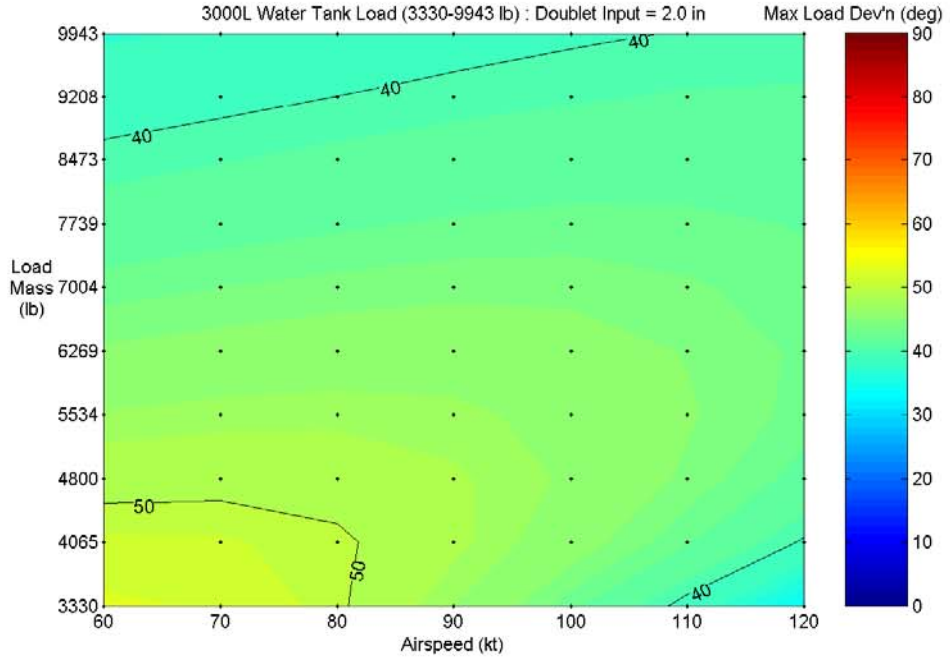

Figure 3.19: Max Load Deviation Contour Plot: CH-47D + 3000 L Water Tank, Lateral Doublet Input Magnitude = 2.0 in

Another script, **hslrng** performs essentially the same operation, but presents the data in a different way, as shown in Figure 3.20. With this script, the maximum load deviation magnitude is plot against airspeed using lines of constant load mass. There is one group of lines for each control input, all drawn on the same graph.

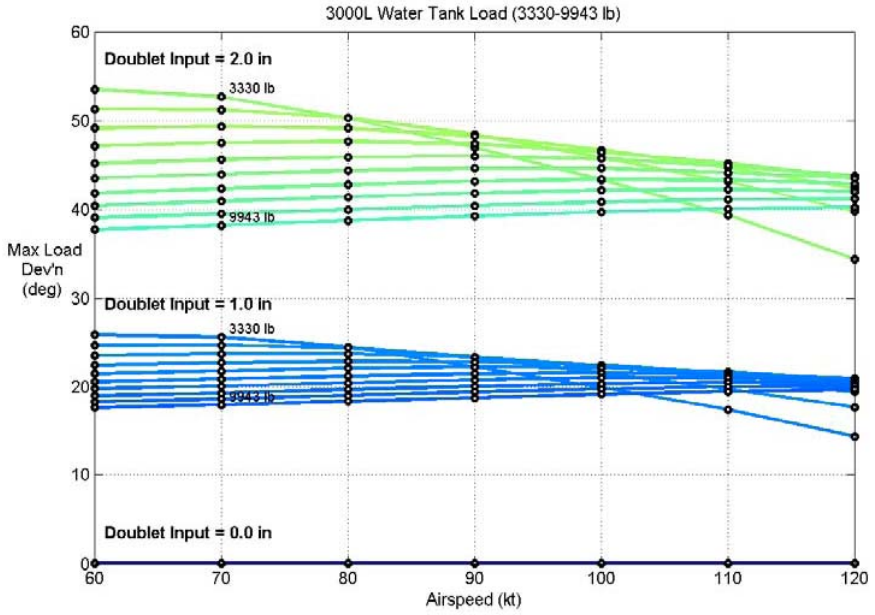

Figure 3.20: Max Load Deviation Range Plot: CH-47D + 3000L Water Tank

Both **hslmap** and **hslrng** will work for most simulation experiments. However, they may need to be edited if different data is to be examined, or the resulting data is not presented in the manner desired. For example, if the lateral cg position is specified as an independent variable in the suite of flight-points, or if the user wishes to fit a polynomial curve to the data prior to display.

Of course, these files are just scripts that load the configuration and output data for each flight-point and then compile the relevant information into matrices which are subsequently displayed in various formats. For each experiment, the files that are saved include: one containing the initial configuration information; one containing the configuration information and output data for *each* flight-point simulated (totalling **B_MN** * **B_VN** * **B_TN**); and one with all of the special and extraneous variables used. In the example above, these are: **ch47b3kltsim-config.mat**, **ch47b3kltsim-V060M333D00.mat** ... **ch47b3kltsim-V120M994D20.mat** and **ch47b3kltsim.mat**.

The first file contains the **Config** struct, the second through to the second last contains both **Config** and **Data** structs and the last contains all variables resident following last iteration of the main loop. Therefore, in order to access the data from any particular flight-point, the user needs simply to load the file via the command:

```
>> load ch47b3kltsim-V090M627D20
```

which brings the **Config** and **Data** structs into the workspace. As detailed below (Appendix A.2.1), **Data** contains fields **time**, **state**, **control**, **modcontrol**, **bodyvel**, **bodyacc**, **cableangle**, **loaddeviation**, and **cableforce**. So, to plot the cable force variation against time, issue the command:

```
>> plot(Data.time,Data.cableforce)
```

As another example, the script **ghsl_ctrl** generates the time-control input matrix, **TD** for a control sequence specified in **Config**, so, to plot the control inputs against time, issue the commands:

```
>> N = Config.simdata.numsteps; t = Data.time; ghsl_ctrl
>> for j = 1:4, subplot(2,2,j), plot(t,TD(:,j+1,1)), end
```

Lastly, it is very easy to create a script similar to **hslmap** and **hslrng** that cycle through the flight-points, load the data from each one, and compile specific values into a large matrix for display. The following commands do just that, for the cable roll (lateral swing) angle over a range of airspeeds.

```
>> B_title = 'ch47b3kltsim'; B_mdir = [pwd,'\HSL\',B_title,'\'];
>> load([B_mdir,B_title]), B_dP = zeros(N,B_VN);
>> for B_j = 1:B_VN
>>    B_i = 1; B_k = 3;
>>    B_mfile = sprintf([B_title,'-V%03dM%03dD%02d'], ...
>>       round(B_V(B_j)),round(B_m(B_i)/10),round(B_T(B_k)*10));
>>    fprintf([B_mfile,'\n']), load([B_mdir,B_mfile])
>>    B_dP(:,B_j) = Data.cableangle(:,1);
>> end
>> surf(B_V,t,B_dP*180/pi); view([62.5,30])
>> xlabel('Airspeed (kt)'), ylabel('Time (s)')
>> title('Cable roll angle (deg)')
```

The result from this script is shown in Figure 3.21.

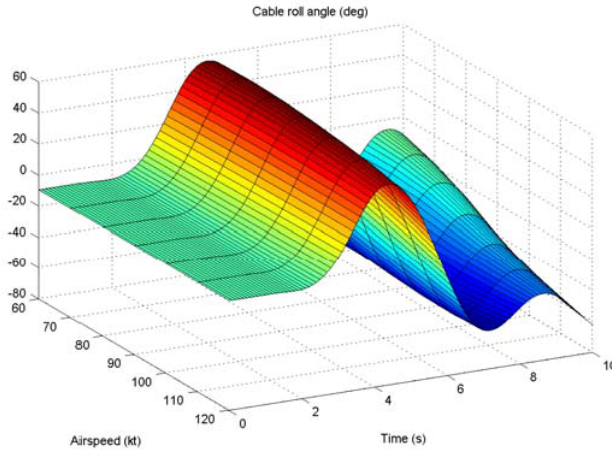

Figure 3.21: Cable Roll Angle: CH-47D + 3000 L Water Tank, Load Mass = 3330 lb; Lateral Doublet Input Magnitude = 2.0 in

Full details of the data matrices are outlined in [20]. For a single-cable configuration, the system state matrix:

$$X = [U \quad R] \tag{4}$$

where the configuration velocity and position matrices:

$$\begin{aligned} U &= [\dot{x}_1 \quad \dot{y}_1 \quad \dot{z}_1 \quad \dot{x}_{c1} \quad \dot{y}_{c1} \quad \dot{z}_{c1} \quad p_1 \quad q_1 \quad r_1 \quad p_2 \quad q_2 \quad r_2] \\ R &= [x_1 \quad y_1 \quad z_1 \quad \phi_1 \quad \theta_1 \quad \psi_1 \quad x_2 \quad y_2 \quad z_2 \quad \phi_2 \quad \theta_2 \quad \psi_2] \end{aligned} \tag{5}$$

and $[\dot{x}_1 \quad \dot{y}_1 \quad \dot{z}_1]$ represent the helicopter inertial velocities, $[\dot{x}_2 \quad \dot{y}_2 \quad \dot{z}_2]$ represent the load inertial velocities, $[\dot{x}_{c1} \quad \dot{y}_{c1} \quad \dot{z}_{c1}]$ are the cable velocities in cable axes, $[\phi_1 \quad \theta_1 \quad \psi_1]$ are the helicopter Euler angles, and $[\phi_2 \quad \theta_2 \quad \psi_2]$ are the load Euler angles. For other configurations, the velocities used in place of the cable velocities are different, but still relative to the helicopter.

The control matrix:

$$C = [\delta_b \quad \delta_a \quad \delta_r \quad \delta_c] \tag{6}$$

where δ_b denotes the longitudinal stick control, δ_a denotes the lateral stick, δ_r denotes the pedals and δ_c denotes the collective stick. Also, the augmented control matrix CC has the same control inputs, modified by any control-feedback commands present.

The body velocity matrix:

$$V_a = [u_1 \quad v_1 \quad w_1 \quad u_2 \quad v_2 \quad w_2 \quad p_1 \quad q_1 \quad r_1 \quad p_2 \quad q_2 \quad r_2] \tag{7}$$

and the body acceleration matrix $\dot{V}_a$ is just the differential of V_a.

The cable angle matrix:

$$A_{cj} = [\phi_{cj} \quad \theta_{cj} \quad \psi_{cj}] \tag{8}$$

comprises the roll, pitch and yaw angles of the first cable (forward-starboard) supporting the load. The load deviation vector A_{jj}, represents the absolute angle made by a line from the helicopter cg to the load cg from its trim orientation.

Lastly, the cable force, FC is a vector of the non-dimensional tensile force within the cable.

All of the data matrices and vectors will normally have the length specified in the simulation configuration. Note, however, that if the simulation has diverged, or run into error, or stopped prematurely, the length of these will be less.

3.7.2 Extending the Model

Aside from writing scripts to run multiple simulations or access the results for display, the user may wish to alter the model itself. Common tasks might involve changing the load's centre-of-gravity location, or adjusting the limits imposed during trim. Some of these aspects will now be discussed.

In order to alter the (j-1)th load's cg position, the only variables that need to be changed are those in the struct **`LDAT_(j).S`**, which holds all the geometric data for the load. Specifically, these are: **`LDAT_(j).S.Links.Data`** and **`LDAT_(j).S.Patches.Data`** — both 3-column matrices with (x, y, z) data for the links (load attachments) and the patches (load surface); their length is unimportant. The positions of the links and patches are both relative to the load's centre-of-gravity, so the cg can be effectively moved by displacing the links and patches by some amount. For example, in the script for the trim and simulation of the CH-47D and ANTPQ-36 Radar, **`ch47bantpqsim.m`**, the cg is offset horizontally by an incremental amount, **`B_x`** with the commands:

```
LDAT_(2).S.Links.Data(:,1) = B_links - B_x(B_i);
LDAT_(2).S.Patches.Data(:,1) = B_patches - B_x(B_i);
```

Note that the links and patches must be reset, and the load position adjusted to account for the offset, before generating a VRML model, since this function assumes that the cg is at the load's centre. It is also possible to change the location of load's attachment points in this way.

To *rearrange* the helicopter-load attachment locations, the cable index matrix for the (j-1)th load **`CDAT_(j).i`** must be edited. This is a 2-row matrix with each column representing a different cable in the configuration. The top row lists the helicopter attachment indices and the bottom row lists the load attachment indices. The attachment points are ordered aftward from the forward-most location, or in a clockwise direction (looking down) from the forward starboard location, as detailed in Section 3.3.2. Typically,

```
CDAT_(j).i = [ ci_heli_fwd_to_aft ; ci_load_cw_from_fwdstbd ];
```

Thus, for a four-cable configuration, the attachment points will be ordered: forward-starboard, aft-starboard, aft-port, forward-port. If those cables are attached in a tandem configuration to the forward and aft attachment points on a helicopter with three hooks, the matrix will be equal to **`[1,3,3,1;1,2,3,4]`**. If the aft two cables are brought forward to the middle helicopter hook, the matrix will become **`[1,2,2,1;1,2,3,4]`**. A bifilar configuration will normally use the same helicopter hooks (forward and aft) as a tandem configuration, so the matrix would be numbered: **`[1,3;1,2]`**. Bringing the aft cable forward would change this to **`[1,2;1,2]`**. If a second, multiple-point load were placed on the third hook, it would have indices equal to **`[3,3,3,3;1,2,3,4]`**.

Of course, the properties of individual cables can also be modified by directly editing any of the fields within the cable data matrix, **`CDAT_`**. The stiffness and damping of the cables are held in vectors **`CDAT_(j).K`** and **`CDAT_(j).C`**, respectively. The unstretched and current lengths are in **`CDAT_(j).l0`** and **`CDAT_(j).l`**, respectively. However, care must be taken if adjusting

these individually, because for tandem configurations, the forward two slings and aft two sling must have the same lengths. For multiple-point configurations, all four slings must have the same length; otherwise the load cannot be set to its initial position, before balancing. The cable stretch rates **CDAT_(j).ldot** and the cable direction vectors **CDAT_(j).k_N**, should not be altered.

The various fields within the helicopter and load data matrices **HDAT_** and **LDAT_** can also be changed directly, but would be wiser to edit the m-files that create them. For the helicopter, these files are **hsl_ch47bdat.m**, **hsl_ch53ddat.m**, and **hsl_uh1hdat.m**. The fields include the linearised aerodynamic coefficients **HDAT_.A**, helicopter reference area **HDAT_.R**, reference length **HDAT_.L**, as well as the mass **HDAT_.m**, inertia **HDAT_.J**, control limits **HDAT_.C**, and geometric structure **HDAT_.S**. For the loads, there is **hsl_airfoildat.m**, **hsl_boxdat.m**, **hsl_cylinderdat.m**, **hsl_milvandat.m**, **hsl_platedat.m**, **hsl_ribdat.m**, and **hsl_sboxdat.m**. The fields created in each include the aerodynamic coefficients **LDAT_(j).A**, mass **LDAT_(j).m**, inertia **LDAT_(j).J**, geometric structure **LDAT_(j).S**. The only field that should not be altered is the load angle vector **LDAT_(j).a**.

One useful experiment involves constraining the helicopter-load model in certain axes during both trim and simulation. This can be achieved with the use of infinite mass and inertial elements in the helicopter and load data matrices. For example, the helicopter and load can be constrained to the x-z plane by setting: **m(2) = Inf**, **J(1,1) = Inf**, and **J(3,3) = Inf** for both **HDAT_** and **LDAT_(j)**. Further, the helicopter can be constrained to move only along the x-axis by also setting: **HDAT_.m(3) = Inf** and **HDAT_.J(2,2) = Inf**. The matrix that is actually used in the functions is the combined helicopter-load mass-inertia matrix **D**, which is created in **ghsl_init** by concatenating each of the sub matrices along the main diagonal.

Finally, it may become necessary to increase the limits imposed during trim if the true angles associated with the helicopter and load orientation are large. These are set in **hsl_trimfun.m** and depend on the sling configuration and the airspeed at trim. The matrix **dlim** contains the upper and lower limits for the helicopter roll and pitch angles and the control inputs. Those control limits actually come from the helicopter data matrix **HDAT_.C**, and are created in the m-files above. The matrix **rlim** defines the upper and lower limits for each load's roll, pitch and yaw angles in an elastic configuration, while **alim** defines the limits for other load angles (that vary according to the sling configuration) used in an inelastic model. Around hover (< 10ft/s), the absolute limits generally range from 5° to 20°, and above that, they range from 10° to 60°. It is possible to see the optimisation routine hitting these limits in the REPLAY display during trim – the helicopter, loads or controls will turn red. The controls are hard-limited, so if the limits are met, they will not be exceeded. All of the angles, on the other hand, have only soft limits that may be exceeded. However, any variable that has reached or passed its limit will substantially penalise the target function, which for trim is the residual (out-of-balance) external force. As a result, the optimisation routine should stay within the limits specified. While it is possible to maintain realistic limits for the helicopter and load angles, it can be much harder to stay within the control limits of the helicopter. If the controls are hitting their limits for a range of different flight-points, it is likely that the helicopter-load system is outside its envelope and will not be able to achieve trim.

4. Conclusion

A system description has been provided for the helicopter slung load simulation software package HSLSIM, followed by a detailed User's Guide. Examples are provided throughout for a CH-47D helicopter carrying a 3000 L water tank and responding to a lateral doublet control input. A description of the simulation procedure is detailed for both GUI and script-driven execution. The model should provide Army personnel with an insight into dynamic stability of slung loads before flight testing thereby providing improved safety and reduced flight testing time.

5. References

[1] R. A. Stuckey, "Dynamic Simulation of the CH-47D Helicopter with Single and Multiple Slung Loads," Proceedings of 2nd Australian Pacific Vertiflite conference on Helicopter Technology, Canberra, ACT, 1998.

[2] L. S. Cicolani, G. Kanning, and R. Synnestvedt, "Simulation of the dynamics of helicopter slung load systems," *American Helicopter Society, Journal*, vol. 40, pp. 44-61, 1995.

[3] L. S. Cicolani and G. Kanning, "General equilibrium characteristics of a dual-lift helicopter system," NASA-TP-2615; A-86114; NAS 1.60:2615, 1986.

[4] *MATLAB®: The Language of Technical Computing. Using MATLAB® Version 6*. Natick, MA: The Mathworks, Inc., 2000.

[5] J. M. Weber, T. Y. Liu, and W. Chung, "A Mathematical Simulation Model of the CH-47B Helicopter, Volume 1," NASA-TM-84351-VOL-1; A-9303-VOL-1; NAS 1.15:84351-VOL-1, 1984.

[6] J. M. Weber, T. Y. Liu, and W. Chung, "A Mathematical Simulation Model of the CH-47B Helicopter, Volume 2," NASA-TM-84351-VOL-2; A-9303-VOL-2; NAS 1.15:84351-VOL-2, 1984.

[7] R. K. Heffley, "ROTORGEN Minimal-Complexity Simulator Math Model with Application to Cargo Helicopters," Hoh Aeronautics, Inc., Lomita, CA, NASA Contractor Report NASA-CR-196705, March 1997.

[8] T. Ronen, *Dynamics of a Helicopter with a Sling Load (PhD Thesis)*. Department of Aeronautics and Astronautics, Stanford University, California, 1986.

[10] S. F. Hoerner, *Fluid dynamic drag: practical information on aerodynamic drag and hydrodynamic resistance*. Midland Park, New Jersey, 1965.

[11] S. F. Hoerner and H. V. Borst, *Fluid dynamic lift: practical information on aerodynamic and hydrodynamic lift*. Brick Town, New Jersey, 1975.

[12] W. Johnson, "A Comprehensive Analytical Model of Rotorcraft Aerodynamics and Dynamics: Theory Manual," vol. 1. Palo Alto, CA: Johnson Aeronautics, 1988.

[13] "Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems," American National Standards Institute, American Institute of Aeronautics and Astronautics, Washington, DC ANSI/AIAA R-004-1992, February 1992.

[14] "Cortona VRML Client," 4.0 ed. Dublin, Ireland: Parallel Graphics, Inc., 2003.

[15] J. Gailly, "gzip (GNU zip)," 1.2.4 ed. Rueil-Malmaison, France, 1992.

[16] *Control System Toolbox. For Use with MATLAB®*. The Mathworks, Inc., 2000.

[17] *Optimization Toolbox. For Use with MATLAB®*. Natick, MA: The Mathworks, Inc., 2000.

[18] R. Carey, G. Bell, and C. Marrin, "The Virtual Reality Modeling Language (VRML) - Parts 1-2: Functional Specification and UTF-8 Encoding and; External Authoring Interface," International Organization for Standardization ISO/IEC 14772-1, 1997.

[19] B. Etkin, *Dynamics of Atmospheric Flight*. New York: Wiley, 1972.

[20] R. A. Stuckey, "Mathematical Modelling of Helicopter Slung-Load Systems," Defence Science and Technology Organisation, Aeronautical and Maritime Research Laboratory, Melbourne, DSTO Technical Report DSTO-TR-1257, December 2001.

Appendix A: MATLAB® Toolbox

A.1. Package Contents

A.1.1 MATLAB® M-files, MEX-files and MAT-files

a2q.m	Angular displacement conversion; Angular→Quaternion representation
alt2sig.m	Altitude to density-ratio conversion
ch47b3kltsim.m	Simulation script for CH-47D + 3000L Water-tank system
ch47bantpqsim.m	Simulation script for CH-47D + ANTPQ-36 Radar system
ch47bmrtfsim.m	Simulation script for CH-47D + Merlo Rough Terrain Forklift system
ch47btclsim.m	Simulation script for CH-47D + Trailer Cargo Lightweight system
count.m	Graphical progress count & time indicator
dtime.m	Time string formatted in years, months, days, hours, mins & secs
e2q.m	Angular displacement conversion; Euler->Quaternion representation
e2v.m	Angular displacement conversion; Euler->VRML rotation representation
erates.m	Euler transform matrix for angular velocities
eratesi.m	Inverse Euler transform matrix for angular velocities
euler.m	Transformation matrix for an Euler-angle triplet
eulers.m	Transformation matrix for an Euler-angle rotation sequence
ghsl_about.m	GHSLSIM function to create 'About... ' dialog window
ghsl_ctrl.m	GHSLSIM function to generate control inputs from config data
ghsl_ctrlcfg.m	GHSLSIM function to create control input scheme
ghsl_helicfg.m	GHSLSIM function to configure helicopter
ghsl_init.m	GHSLSIM function to initialise trim/simulation with config data
ghsl_loadcfg.m	GHSLSIM function to configure load(s)
ghsl_plot.m	GHSLSIM function to plot output from simulation
ghsl_prefs.m	GHSLSIM function to set program preferences
ghsl_triminit.m	GHSLSIM function to set initial trim state
ghslsim.m	Graphical interface to the Helicopter Slung-load Simulation program
hsl_airfoildat.m	Airfoil (wing) aerodynamic, mass, inertial and geometric data
hsl_bifilar.m	Determine load position for bifilar sling configuration
hsl_boxdat.m	Box container aerodynamic, mass, inertial and geometric data
hsl_ch47bdat.m	CH-47B aircraft aerodynamic, mass, inertial and geometric data
hsl_ch53ddat.m	CH-53D aircraft aerodynamic, mass, inertial and geometric data
hsl_checkopts.m	Check input parameters for HSLSIM
hsl_coeffs.m	Create the aerodynamic state-space matrices
hsl_config.m	Create the configuration matrices and the basis matrix for solution
hsl_cylinderdat.m	Cylindrical container aerodynamic, mass, inertial and geometric data
hsl_forces.m	Compute the accelerations for a helicopter slung-load system
hsl_indices.m	Generate subsystem state vector indices for one body
hsl_loadforces.m	Compute the aerodynamic force for one load
hsl_loadfun.m	Minimisation function calculates load position & orientation for balance
hsl_milvandat.m	MILVAN container aerodynamic, mass, inertial and geometric data
hsl_output.m	Output display function for HSL_FORCES

hsl_platedat.m	Flat plate aerodynamic, mass, inertial and geometric data
hsl_ribdat.m	Rigid Inflatable Boat aerodynamic, mass, inertial and geometric data
hsl_sboxdat.m	Small box Container aerodynamic, mass, inertial and geometric data
hsl_setload.m	Calculate the load position from the helicopter position and orientation as well as various load and cable angles
hsl_trimfun.m	Minimisation function to calculate helicopter and load position and orientation for trim
hsl_uh1hdat.m	UH-1H aircraft aerodynamic, mass, inertial and geometric data
hslload.m	Determine the load position through optimisation
hslmap.m	Colour map graphs of max load deviation from multiple HSLSIM runs
hslplot.m	Time-history plots of states and controls from simulation
hslrng.m	Line plots of maximum load deviation from multiple HSLSIM runs
hslsim.m	Helicopter Slung-Load Simulation
hsltrim.m	Determine the helicopter and load trim state through optimisation
hslvrml.m	Generate dynamic VRML model for replay of HSLSIM simulation
j2e.m	Convert j-unit vectors to wing-oriented Euler angles used in HSLVRML
jacobian.m	Numerically compute the Jacobian dF/dY of function F(T,Y)
k2e.m	Convert k-unit vectors to cable-oriented Euler angles
labelfix.m	Fix rotation and alignment settings for y-labels on axes
legendtrans.m	Place transposed legend on current window
linspace2.m	Linearly spaced matrix.
odef.m	Differential equation solver based on Runge-Kutta integration with fixed step-size
pendfreq.m	Estimate slung load frequencies using equivalent pendulum system
q2e.m	Angular displacement conversion; Quaternion -> Euler representation
q2v.m	Angular displacement conversion; Quaternion -> VRML rotation
qratesi.m	Inverse quaternion transform matrix for angular velocities
quaternion.m	Transformation matrix for a quaternion set
replay.m	Display dynamic system response path
setportrait.m	Set current paper positions for portrait orientation
skew3.m	General skew-symmetric matrix S(x,y,z)
subplot_.m	Create axes in tiled positions with strict scaling
t2e.m	Convert transformation matrix to Euler angles
v2a.m	Calculate quaternion transform defined by two vectors

mexrotorgen.dll	Compute aerodynamic forces using RotorGen MEX model
ghsldefault.mat	Default configuration file
ghsldemo_*.mat	Demonstration configuration files with the CH-47D helicopter
ghsldemo_3klt.mat	3000L Water-tank load; Multiple-point slings
ghsldemo_3klt_trim.mat	As above with stable trim estimate
ghsldemo_3klt_run.mat	As above with simulation data
ghsldemo_bifilar.mat	Box container; Bifilar slings
ghsldemo_multiload.mat	Two Box containers; Single and multiple-point slings
ghsldemo_multiple.mat	Box container; Multiple-point slings
ghsldemo_noload.mat	No load (helicopter only)
ghsldemo_single.mat	Box container; Single-point sling
ghsldemo_single_trim.mat	As above with stable trim estimate
ghsldemo_single_run.mat	As above with simulation data
ghsldemo_tandem.mat	Box container; Tandem slings
ghsldemo_tandem_trim.mat	As above with stable trim estimate
ghsldemo_tandem_run.mat	As above with simulation data
ghslprefs.mat	Program preferences file

A.1.2 VRML Model Files

antpq.wrl	ANTPQ-36 Radar load
box.wrl	Generic rectangular load
ch47b.wrl	CH-47B Chinook Helicopter with two main rotors
ch47b-tex.wrl	As above with image-based texture mapping
cylinder.wrl	Generic cylindrical load
ground.wrl	Ground terrain mesh with random height and texture mapping
mms.wrl	MMS Medium Maintenance Shelter load
mms-tex.wrl	As above with image-based texture mapping
plate.wrl	Generic plate load
rib.wrl	RIB Rigid Inflatable Boat load
s70a9.wrl	S70-A9 Blackhawk Helicopter with main and tail rotors
tcl.wrl	TCL Trailer Cargo Lightweight load
uh1h.wrl	UH-1H Iroqois Helicopter with main and tail rotors

A.2. Code Listing: Multiple Simulations

A.2.1 ch47b3kltsim.m

```
% CH47B3KLTSIM : Simulation script for CH-47D + 3000L Water-tank system
%
%                CH47B3KLTSIM
%
%                Edit for details ...

% R.A. Stuckey  20/01/03  (c) 2003, Defence Science and Technology Organisation
% ------------------------------------------------------------------------------

% CONFIGURATION SECTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check if script has been run before during the same session
if ~exist('B__','var')

% First, the title and directory used for all saved files must be set
   B_title = 'ch47b3kltsim';
   B_mdir = [pwd,'\HSL\',B_title,'\'];
   if ~exist(B_mdir), mkdir(B_mdir); end

% As well as the range of load masses used
   B_m = linspace(3330,9943,10);
% And the range of airspeeds at which the helicopter is trimmed
   B_V = [60:10:120];
% Also, the control input magnitudes
   B_T = [0.0,1.0,2.0];

% Next, the preferences associated with ghslsim need to be specified
   Prefs = struct( ...
      'webbrowser','', ...
      'vrmldir', [pwd,'\..\VRML\HSL\'], ...
      'gzipexe','C:\Program Files\WinZip\gzip.exe', ...
      'replaywin',1, ...
      'vtrackwin',1, ...
      'countwin',1, ...
      'verbose',1 ...
   );

% Create the configuration struct, which contains all of the remaining data for
% the simulation, helicopter, load, controls and trim.
   Config = struct( ...
      'simdata',struct( ...
         'inttime',      0.100, ...
         'numsteps',     101, ...
         'simaxes',      'combined', ...
         'angrep',       'euler', ...
         'simtype',      'nonlinear', ...
         'slingmod',     'elastic', ...
         'trimfun',      'lsqnonlin', ...
         'simfun',       'ode45', ...
         'trimvel',      [B_V(1)/0.5925,0,0], ...
         'trimalt',      100.0, ...
         'numloads',     1 ...
      ), ...
      'helidata',struct( ...
         'helitype',     'ch47', ...
         'simmodel',     'mex', ...
         'scas',         0, ...
         'heliweight',   46000, ...
         'heliinertia', [43200,0,0;0,250000,0;0,0,234000] ...
      ), ...
      'loaddata',struct( ...
         'loadtype',     {'box'}, ...
         'loadweight',   {B_m(1)}, ...
         'loadinertia', {diag([ 0.33 1.20 1.20 ]*B_m(1))}, ...
         'loadsize',     {[96.0,77.5,46.0]/12}, ...
         'slingcfg',     {'multiple'}, ...
         'cablelength', {16.0 + 3.5}, ...
         'cablestiff',  {20000.0}, ...
         'cabledamp',   {500.0}, ...
         'attachoffset',{[1,1,-1;-1,1,-1;-1,-1,-1;1,-1,-1]*diag([96.0,77.5,46.0]/2/12)}, ...
         'cableslack',  {0} ...
      ), ...
      'ctrldata',struct( ...
```

```
        'inputtype',    {{'ratefb'},{'doublet','pulse'},{'doublet'},{'none'}}, ...
        'starttime',    {{0},{1.0,4.5},{1.0},{0}}, ...
        'ramptime',     {{0},{0.5,0.5},{0.5},{0}}, ...
        'holdtime',     {{10},{1.0,0.5},{1.0},{0}}, ...
        'magnitude',    {{0},{B_T(1),-B_T(1)},{B_T(1)*0.5},{0}}, ...
        'rategain',     {{40},{0,0},{0},{0}} ...
     ), ...
     'trimdata',struct( ...
        'r0',            [0,0,0,0,0,0,0,5*pi/180,0,0,0,0]', ...
        'd0',            [0,0,0,4.56], ...
        'xt',            zeros(1,12) ...
     ) ...
  );

  save([B_mdir,B_title,'-config'],'Config')

% Lastly, specify the VRML geometric helicopter and load models to be utilised
  wrlfiles = { [], [Prefs.vrmldir,'ch47b.wrl'], [Prefs.vrmldir,'box.wrl'] };

% END CONFIGURATION SECTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  global opt_ pref_

% Create the main index matrix which specifies the order of simulation runs
  B_mN = length(B_m); B_VN = length(B_V); B_TN = length(B_T);
  reshape([1:B_mN*2*ceil(B_VN/2)],B_mN*2,ceil(B_VN/2)); ans([1:B_mN,B_mN*2:-1:B_mN+1],:);
  B_I = reshape(ans(1:B_mN*B_VN),B_mN,B_VN); B_I2 = zeros(B_mN,B_VN);

% Create some more special variables used in the main loop of the script
  n = Config.simdata.numloads + 1;
  B_dA = zeros(B_mN,B_VN,B_TN); B__ = B_dA;
  B_u0  = zeros(n*6,B_mN,B_VN); B_r0  = B_u0; B_xt = B_u0; B_d0  = zeros(4,B_mN,B_VN);

  [B_i,B_j] = find(B_I == 1);

% Set the position & orientation vector, the control vector and the trim state
% vector for the first step
  B_r0(:,B_i,B_j) = Config.trimdata.r0;
  B_d0(:,B_i,B_j) = Config.trimdata.d0';
  B_xt(:,B_i,B_j) = Config.trimdata.xt';

else

  fprintf('\n Resuming script ...\n')
end

% Set some globals used by hsltrim and hslsim in each simulation run
global HDAT_ LDAT_ CDAT_

B_TR = cputime;

% Initialise the hsltrim variables
ghsl_init

B_links = LDAT_(2).S.Links.Data(:,3);
B_patches = LDAT_(2).S.Patches.Data(:,3);

% Start the main loop
for B_ii = 1:B_mN*B_VN

% Identify the current airspeed and load mass vector indices
  [B_i,B_j] = find((B_I == B_ii)&(B__(:,:,1) < 2));

% Skip this step if the model has not been trimmed and simulated at this point
  if ~isempty(B_i)

% Adjust the load mass and inertia, and the trim velocity for this step
     Config.loaddata(1).loadweight = B_m(B_i);
     Config.loaddata(1).loadinertia = diag([ 0.33 1.20 1.20 ]*B_m(B_i));
     Config.simdata.trimvel(1) = B_V(B_j)/0.5925;

% Display the current trim step
     if (B__(B_i,B_j,1) < 1)
        fprintf('\n##### V = %5.1fkn  m = %4.0flb
##################################################\n',B_V(B_j),B_m(B_i));
     end

% Set the initial trim index either to the current, or to the previous value
     if B_ii == 1
        B_ii2 = B_ii; B_i2 = B_i; B_j2 = B_j; B_I2(B_i,B_j) = B_ii2;
```

```
      else
         B_ii2 = B_ii-1; [B_i2,B_j2] = find(B_I == B_ii2); B_I2(B_i,B_j) = B_ii2;
      end

% Loop while the system does not trim
      while (B__(B_i,B_j,1) < 1)

% Each time, setting the initial trim variables to previous values
         Config.trimdata.r0 = B_r0(:,B_i2,B_j2);
         Config.trimdata.d0 = B_d0(:,B_i2,B_j2)';
         Config.trimdata.xt = B_xt(:,B_i2,B_j2)';

% BALANCE LOADS: INELASTIC CONFIGURATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set the sling model to inelastic
         Config.simdata.slingmod = 'inelastic';

% Initialise the hsltrim variables
         ghsl_init

% Calculate and adjust the load's vertical cg location
         zcg = (25.0 - (23.0*B_m(1) + 23.0*(B_m(B_j)-B_m(1))^2/(B_m(end)-B_m(1)))/B_m(B_j))/12;
         LDAT_(2).S.Links.Data(:,3) = B_links - zcg;
         LDAT_(2).S.Patches.Data(:,3) = B_patches - zcg;

         if pref_.verbose, fprintf('\n Balancing loads (inelastic config) ... \n'), end

% Set the initial load position directly beneath the fixed helicopter
         jj = 3+[1:3];
         if ~any(r0([jj,n*3+jj]))
            r0([jj,n*3+jj]) = hsl_setload(u0,r0,[],j);
         end

% Bring up the replay window if requested
         if Prefs.replaywin
            hr = replay(0,r0(1:n*3)'-r0([1:3]'*ones(1,n),1)',r0(n*3+[1:n*3])',[],s);
         end

% Balance the load beneath the fixed helicopter
         hslload

% Update the load positional trim data
         Config.trimdata.r0 = r0;

% BALANCE LOADS: ELASTIC CONFIGURATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set the sling model to elastic
         Config.simdata.slingmod = 'elastic';

% Initialise the hsltrim variables
         ghsl_init

         if pref_.verbose, fprintf('\n Balancing loads (elastic config) ... \n'), end

% Set the initial load position directly beneath the fixed helicopter
         jj = 3+[1:3];
         if ~any(r0([jj,n*3+jj]))
            r0([jj,n*3+jj]) = hsl_setload(u0,r0,[],j);
         end

% Bring up the replay window if requested
         if Prefs.replaywin
            hr = replay(0,r0(1:n*3)'-r0([1:3]'*ones(1,n),1)',r0(n*3+[1:n*3])',[],s);
         end

% Balance the load beneath the fixed helicopter
         hslload

% Update the load positional trim data
         Config.trimdata.r0 = r0;

% TRIM HELICOPTER + LOADS (ELASTIC CONFIG) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialise the hsltrim variables
         ghsl_init

% Bring up the replay window if requested
         if Prefs.replaywin
            hr = replay(0,r0(1:n*3)'-r0([1:3]'*ones(1,n),1)',r0(n*3+[1:n*3])',[],s);
         end
```

```
% Set the trim time-control matrix to it's initial value
        TD0 = zeros(1,5,2); TD0(1,2:5,1) = d0;

% Trim the helicopter load system
        hsltrim

% Update the trim control vector
        d0 = TD0(1,2:5,1);

% Check the residual (out-of-balance) external force sum against some tolerance
        if (ff > 1e-1)
% Trim point not reached
           if B_ii2 == 1
% No earlier trim points remaining; Exit the script
              fprintf('\n')
              error(sprintf('Large error: F = %g  N = %g; Exiting ...',ff,Bii));
           else
% Try earlier trim point for initial values
               B_ii2 = B_ii2-1; [B_i2,B_j2] = find(B_I == B_ii2); B_I2(B_i,B_j) = B_ii2;

              fprintf('\n')
              warning(sprintf('Large error: F = %g  N = %g; Attempting re-trim from V = %5.1fkn  m =
%4.0flb',ff,B_ii,B_V(B_j2),B_m(B_i2)));
              fprintf('\n')
           end
% Flag the corresponding special matrix element to denote an untrimmed state
           B__(B_i,B_j,:) = 0;
        else
% Trim point successfully reached - update the special matrices for the
% configuration velocity, position & orientation, controls and trim state
           B_u0(:,B_i,B_j) = u0; B_r0(:,B_i,B_j) = r0;
           B_d0(:,B_i,B_j) = d0'; B_xt(:,B_i,B_j) = XT';

           Config.trimdata.r0 = r0;
           Config.trimdata.d0 = d0;
           Config.trimdata.xt = XT;

% Flag the corresponding special matrix element to denote a trimmed state
           B__(B_i,B_j,:) = 1;
        end
     end

% Find the simulation cases that have yet to be run for the current trim point
     B_K = find(B__(B_i,B_j,:) < 2)';

% And loop through those
     for B_k = B_K

% Update the control magnitudes
        Config.ctrldata(2).magnitude = {B_T(B_k),-B_T(B_k)};
        Config.ctrldata(3).magnitude = {B_T(B_k)*0.5};

% Display the current simulation step
        fprintf('\n##### D = %3.1fin\n',B_T(B_k));

% SIMULATE HELICOPTER + LOADS (ELASTIC CONFIG) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialise the hslsim variables
        ghsl_init

% Bring up the replay window if requested
        if Prefs.replaywin
           hr = replay(0,r0(1:n*3)'-r0([1:3]'*ones(1,n),1)',r0(n*3+[1:n*3])',[],S);
        end

% Set the trim time-control matrix to its initial value
        TD0 = zeros(1,5,2); TD0(1,2:5,1) = d0;

% Set the configuration velocity vector, the position & orientation vector and
% the full state vector to their initial values
        u = u0; r = r0; x = [u;r];

% Generate the control inputs for the specified time sequence
        ghsl_ctrl

% Offset the time-control matrix by its trim value
        TD(:,2:5,1) = TD(:,2:5,1)+ones(N,1)*TD0(1,2:5,1);

% Run the simulation
```

```
        hslsim

% Close the replay window if present
        if Prefs.replaywin
           close(hr)
        end

% Extract the (open and closed loop) time-control input matrices
        C = TD(:,2:5,1); CC = TDD(:,2:5);

% Create a struct with the simulation output
        Data = struct('time',t, ...
                      'state',X, ...
                      'control',C, ...
                      'modcontrol',CC, ...
                      'bodyvel',Va, ...
                      'bodyacc',Vadot, ...
                      'cableangle',Acj, ...
                      'loaddeviation',Ajj, ...
                      'cableforce',FC);

% Append the m-file name with airspeed, load mass and control magnitude fields
        B_mfile = sprintf([B_title,'-
V%03dM%03dD%02d'],round(B_V(B_j)),round(B_m(B_i)/10),round(B_T(B_k)*10));

% Save the Config and Data structs to file
        save([B_mdir,B_mfile],'Config','Data')

% Update the maximum load deviation special matrix
        B_dA(B_i,B_j,B_k) = max(abs(Ajj(:,2)));

% GENERATE VRML MODEL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reset the load's vertical cg location to centre - necessary for hslvrml
        LDAT_(2).S.Links.Data(:,3) = B_links;
        LDAT_(2).S.Patches.Data(:,3) = B_patches;

% Adjust the load position history to account for the new cg location
        for i = 1:N
           X(i,n*6+3+[1:3]) = X(i,n*6+3+[1:3])-[0,0,zcg]*euler(X(i,n*9+3+[1:3]));
        end

% Fill the first element of wrlfiles with the output filename
        wrlfiles{1} = [Prefs.vrmldir,B_title,'\',B_mfile,'.wrl'];

% Define the user and fixed-inertial viewpoint offsets
        rv = [ 75 75 0 ];
        dX = X - ones(size(X,1),1)*X(1,:);
% Place fixed-inertial at midpoint + offset, but above ground
        ri = (min(dX(:,n*6+[1:3])) + max(dX(:,n*6+[1:3])))/2 + [ 0 100 100 ];
        if ri(3) > (-X(1,n*6+3) - 5), ri(3) = -X(1,n*6+3) - 5; end

        if pref_.verbose, fprintf(' Generating VRML ... \n\n'), end

% Generate the VRML model
        hslvrml(wrlfiles{:},t,X(:,n*6+[1:n*6]),C,1,pi/4,rv,ri)

% If gzip exists, compress the VRML file
        if ~isempty(Prefs.gzipexe)
           [s,w] = system(['"',Prefs.gzipexe,'" "',Prefs.vrmldir,B_title,'\',B_mfile,'.wrl" -c >
"',Prefs.vrmldir,B_title,'\',B_mfile,'.wrz']);
           if isempty(w), delete([Prefs.vrmldir,B_title,'\',B_mfile,'.wrl']); end
        end

% Adjust the load's vertical cg location again
        LDAT_(2).S.Links.Data(:,3) = B_links - zcg;
        LDAT_(2).S.Patches.Data(:,3) = B_patches - zcg;

% Flag the corresponding special matrix element to denote a simulated state
        B__(B_i,B_j,B_k) = 2;
     end

% Append the cputime
     B_TR = [B_TR,cputime];

     if B_ii < B_mN*B_VN
% Estimate and display the time remaining until completion
        fprintf(' Progress: %g/%g  Time remaining: %s\n\n', ...
                B_ii,B_mN*B_VN,dtime((B_mN*B_VN-B_ii)*mean(diff(B_TR))))
     else
```

```
% Display the total execution time
         fprintf(' Total execution time: %s\n\n',dtime(B_TR(end)-B_TR(1)))
      end
   end
end

% Save all variables, including the special variables to file
save([B_mdir,B_title])
```

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
Helicopter Slung-Load Simulation Toolbox for use with MATLAB®	Document (U) Title (U) Abstract (U)

4. AUTHOR(S)	5. CORPORATE AUTHOR
Roger A. Stuckey	DSTO Defence Science and Technology Organisation 506 Lorimer St Fishermans Bend Victoria 3207 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TN-0855	AR-014-325	Technical Note	August 2008

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
2003/70263	ARM 07/038	ARMY	49	20

13. URL on the World Wide Web	14. RELEASE AUTHORITY
http://www.dsto.defence.gov.au/corporate/reports/DSTO-TN-0855.pdf	Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS http://web-vic.dsto.defence.gov.au/workareas/library/resources/dsto_thesaurus.shtml

Chinook helicopter; Slung load; Modelling; Simulation

19. ABSTRACT

This document outlines the Helicopter Slung Load Simulation (HSLSIM) Toolbox which is a set of utilities for the simulation, analysis and display of the flight-dynamic response of helicopters with various external load configurations within the MATLAB® software environment. Instructions and examples are provided for its operation and subsequent modifications. The helicopter studied is the CH-47D, with the load types including rectangular and cylindrical containers as well as plate and airfoil shapes.